Scalable Graph Condensation with Evolving Capabilities

Shengbo Gong*
shengbo.gong@emory.edu
Emory University
Atlanta, Georgia, USA

Mohammad Hashemi*
mohammad.hashemi@emory.edu
Emory University
Atlanta, Georgia, USA

Juntong Ni
juntong.ni@emory.edu
Emory University
Atlanta, Georgia, USA

Carl Yang
j.carlyang@emory.edu
Emory University
Atlanta, Georgia, USA

Wei Jin
wei.jin@emory.edu
Emory University
Atlanta, Georgia, USA

Abstract

The rapid growth of graph data creates significant scalability challenges as most graph algorithms scale quadratically with size. To mitigate these issues, Graph Condensation (GC) methods have been proposed to learn a small graph from a larger one, accelerating downstream tasks. However, existing approaches critically assume a static training set, which conflicts with the inherently dynamic and evolving nature of real-world graph data. This work introduces a novel framework for continual graph condensation, enabling efficient updates to the distilled graph that handle data streams without requiring costly retraining. This limitation leads to inefficiencies when condensing growing training sets. In this paper, we introduce GECC (**G**raph **E**volving **C**lustering **C**ondensation), a scalable graph condensation method designed to handle large-scale and evolving graph data. GECC employs a traceable and efficient approach by performing class-wise clustering on aggregated features. Furthermore, it can inherit previous condensation results as clustering centroids when the condensed graph expands, thereby attaining an evolving capability. This methodology is supported by robust theoretical foundations and demonstrates superior empirical performance. Comprehensive experiments including real world scenario show that GECC achieves better performance than most state-of-the-art graph condensation methods while delivering an around 1000× speedup on large datasets.

CCS Concepts

• **Computing methodologies** → **Machine learning**; **Neural networks**.

Keywords

Graph Neural Networks, Graph Condensation, Graph Clustering, Data-Efficient Learning

*Both authors contributed equally to this research.

Table 1: Condensation time on evolving graphs, reported as a multiple of GNN training time (21.37 s) on uncondensed graphs (*Reddit*, 232,965 nodes) in the final time step. The GNN training time on condensed graphs is 1.1 s.

Method	T_1	T_2	T_3	T_4	T_5
GCondX	75×	112×	149×	104×	185×
GCondX-Init	75×	89×	107×	120×	134×
GCondX-Grow	75×	94×	112×	101×	119×

1 Introduction

Graph-structured data has become indispensable in various domains, including social networks [5], epidemiology [24], and recommendation systems [37]. The ability of graphs to represent complex relationships and dependencies has propelled their adoption in machine learning, especially with the advent of graph neural networks (GNNs) [3]. However, the exponential growth of real-world graph datasets presents significant computational challenges, as the cost of training GNNs increases with the number of nodes and edges [15, 17]. To address this, graph condensation (GC) techniques [18, 19, 34, 38, 41, 42] have been developed, which aim to produce significantly smaller yet information-rich graphs that accelerate GNN training. For example, GCond [19] condenses the Flickr dataset to 0.1% of its original size while preserving 99.8% of the original accuracy on node classification. These condensed graphs not only save significant storage space and transmission bandwidth, but also enhance the efficiency of retraining neural networks in many critical applications such as continual learning [23].

Despite the promise, existing GC methods face three significant limitations that limit their practical utility in real-world scenarios.

(a) High Computational Overhead: Most GC methods entail a computationally intensive condensation process, marked by repeated gradient calculations and updates. This process closely resembles full GNN training, creating a paradox as it contradicts the primary objective of GC methods to enhance the efficiency of GNN training. For instance, gradient matching techniques [18, 19] require consistent alignment of GNN gradients across successive iterations. Similarly, trajectory matching methods [41, 42] necessitate the alignment of GNN parameters at various stages during

KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

Shengbo Gong, Mohammad Hashemi, Juntong Ni, Carl Yang, and Wei Jin

the training process. These methods inherently demand significant GNN training and gradient processing, thus slowing down the procedure as the size of graph datasets expands.

(b) Challenges with Evolving Graphs: Existing GC methods are designed for static graphs and overlook the evolving nature of real-world graphs. In practice, graphs evolve over time, with nodes, edges, or attributes being added, removed, or modified. Existing graph condensation (GC) methods require re-running the entire condensation process from scratch whenever the training set changes, since modifications affect every synthetic node. Moreover, while synthetic graphs are expected to grow proportionally with the original graphs, an effective strategy for adapting to this growth is currently lacking. In Table 1, we report the relative condensation times for evolving graphs on the *Reddit* dataset, using GCondX [19] as a representative example. We compare three variants: (i) **GCondX**: Performs condensation from scratch at each time step. (ii) **GCondX-Init**: Retains synthetic nodes from the previous time step as initialization, updating all nodes during the condensation process. (iii) **GCondX-Grow**: Freezes the synthetic nodes from the previous time step and concatenate new synthetic nodes to grow the condensed graph. The table indicates that the condensation process requires over 100× more time than the GNN training on the uncondensed graph. This substantial computational overhead suggests that current GC methods may not be practical for evolving graphs. Even though simple evolving strategies can reduce marginal time, the cost remains significantly higher than that of GNN training, underscoring the urgent need for a novel GC method that can efficiently handle graph evolution.

(c) Lack of Traceability: Many GC methods synthesize condensed graphs without explicitly establishing a connection between the nodes in the condensed graph and those in the original graph. This lack of explicit connections obscures the contribution of each original node to each condensed node, diminishing the traceability of the condensation process. Although these condensed nodes might be informative to GNN models, discerning their real-world semantic meanings can be challenging to humans. This poor traceability can significantly limit the application of GC in sectors where clarity, comprehensive data explanations, and transparent decision-making are crucial. Also, with traceability, users can manipulate the condensed graph alongside the original one—for example, easily filtering low-quality data when the corresponding original nodes exhibit poor quality.

To tackle the three critical challenges all together, we introduce a novel framework called Graph Evolving Clustering Condensation (GECC). We begin by refining our objectives for graph condensation and establishing theoretical upper bounds that clarify the connection between graph condensation and clustering. This connection enables us to perform condensation efficiently without the costly gradient calculations needed for traditional GNN training. Specifically, we cluster propagated node features into partitions, where centroids serve as the condensed node features at each time step. Additionally, this method adapts well to evolving, expanding graphs through incremental clustering, using centroids from previous time steps as initializations for subsequent steps. Notably, GECC enhances traceability by establishing a clear correspondence between condensed and original nodes, offering deeper insights

into how condensed nodes encapsulate information from the actual graph. Our contributions are outlined as follows:

- We provide a novel theoretical understanding of objectives in graph condensation from both the training and test perspectives. Our analysis reveals the connection between graph condensation and clustering, demonstrating that condensation can be effectively addressed by performing clustering with balanced clusters.
- Leveraging our theoretical insights, we introduce the first framework designed to significantly accelerate condensation and adapt to the evolving nature of real-world graphs, while providing traceability. This framework methodically partitions node representations to produce cluster centroids at each time step, which serve as the condensed node features.
- Comprehensive experiments demonstrate that GECC achieves state-of-the-art (SOTA) accuracy and efficiency in GC. For example, GECC can condense the evolving *Reddit* dataset more than 1000 times faster than GCond, while outperforming other existing methods in terms of accuracy.

2 Related Work

2.1 Model-based Graph Condensation

Model-based GC methods, which require a GNN training phase on the original graph, were the first approaches proposed for GC. In these methods, a smaller graph is synthesized to effectively represent the original for training GNNs [14]. For example, GCond [19], DosCond [18], and SGDD [39] minimize a gradient matching loss that aligns the gradients of the training losses w.r.t. the GNN parameters computed on both the original and condensed graphs. Alternatively, SFGC [42] and GEOM [41] condense a graph by aligning the parameter trajectories of the original graph, thereby eliminating the need for explicit edge generation and reducing complexity. Although these approaches are effective, they require significant computational resources due to the multiple full GNN training runs on the original graph. To mitigate this cost, GCDM [22] and SimGC [38] adopt a distribution matching strategy by aligning sampled original node features with synthetic nodes, thus reducing the discrepancy between gradient matrices and the final output. Other methods such as GCSNTK [34] use GNTK [4] to bypass the gradient update in the synthetic graph; however, such approaches currently show suboptimal performance according to recent benchmarks [10, 33]. OpenGC [8] enhances graph condensation for evolving graphs by simulating structure-aware distribution shifts to extract temporal invariant patterns, thereby improving the generalization of GNNs in real-world scenarios. Overall, these model-based GC methods share common challenges including high computational overhead, scalability, and traceability issues. The recently proposed MCond [7] explicitly learns a one-to-many node mapping from original nodes to synthetic nodes, thereby facilitating inductive representation learning and enhancing traceability; however, it still operates within the framework of gradient matching and remains highly complex.

2.2 Model-Agnostic Graph Condensation

While model-based approaches have advanced GC, they still struggle to scale to large-scale graph datasets (e.g., graphs with over one million nodes such as Ogbn-products [15]). To further reduce

Scalable Graph Condensation with Evolving Capabilities

KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

condensation time on such large-scale graphs, recent efforts have focused on **directly matching the representations** between the original and synthetic graphs, bypassing the need for a surrogate model, i.e., the GNN model used during the condensation. This strategy offers benefits in efficiency and generalizability to diverse downstream models. GCPA [21] first extracts structural and semantic information from the original graph, randomly samples the representations, and finally refines them using contrastive learning. Likewise, CGC [9] reduces the representation matching problem to a class partitioning task. In this approach, nodes within the same partition are merged, and a learning module is employed to weight different samples before they are aggregated into a synthetic node. **However**, despite its effectiveness, CGC's theoretical analysis does not establish a relationship between class partitioning and the GC objective, thereby questioning the necessity of this extra step. **Moreover**, these model-agnostic methods still require a *learning module*, which introduces challenges in optimization and hyperparameter tuning. **Finally**, none of these methods fully address the evolving nature of large-scale graphs, which typically necessitates a complete re-condensation whenever the graph changes. *To our best knowledge, GECC is the first model-agnostic and training-free method with linear complexity.*

3 Preliminaries and Notations

Given a node set $\mathcal{V}$ and an edge set $\mathcal{E}$, a graph is denoted as $G = (\mathcal{V}, \mathcal{E})$. In the case of attributed graphs, where nodes are associated with features, the graph can be represented as $G = (\mathbf{X}, \mathbf{A})$, where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$ denotes the node attributes, and $\mathbf{A}$ denotes the adjacency matrix. The graph Laplacian matrix is $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal degree matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. Let $N = |\mathcal{V}|$ and $E = |\mathcal{E}|$ represent the number of nodes and edges, respectively.

3.1 Graph Condensation.

GC aims to create a smaller synthetic graph $G' = (\mathbf{X}', \mathbf{A}')$, where $\mathbf{X}' \in \mathbb{R}^{N' \times d}$, $\mathbf{A}' \in \{0, 1\}^{N' \times N'}$, and $N' \ll N$, from the original large graph $G = (\mathbf{X}, \mathbf{A})$. The objective is to ensure that GNNs trained on G' achieve performance comparable to those trained on G, thereby significantly accelerating GNN training [19]. The large-scale graph $G = (\mathbf{X}, \mathbf{A})$ serves as the original graph of our GC framework. Each node is annotated with one of c classes, encoded as numeric labels $\mathbf{y} \in \{1, \ldots, c\}^N$ and one-hot labels $\mathbf{Y} \in \mathbb{R}^{N \times c}$. Any GC method focuses on finding an update function $\mathcal{F}$ that generates a new condensed graph $G' = \mathcal{F}(G)$, preserving the key structural and feature information required for downstream tasks.

3.2 Evolving Graph Condensation.

Motivation: Training GNNs on real-world graphs is challenging because these graphs constantly evolve, leading to significant computational costs. Current GC techniques, which create smaller training graphs, are designed for static snapshots. Consequently, adapting to graph evolution requires re-running the entire expensive condensation process at each step. This wasteful approach motivates our work on Evolving Graph Condensation, which aims to efficiently update a condensed graph incrementally rather than rebuilding it from scratch. In the evolving graph scenario, we consider a sequential stream of graph batches $\{B_1, B_2, \ldots, B_m\}$, where m represents

the total number of time steps. Each graph batch $B_i = (\mathbf{X}_i, \mathbf{A}_i)$ contains newly added nodes along with their associated edges in *inductive* graphs, while in *transductive* graphs, it contains newly labeled nodes but retains the entire graph structure [32].

Problem Formulation: In the evolving graph scenario, we consider a sequential stream of graph batches $\{B_1, B_2, \ldots, B_m\}$, where m is the total number of time steps. Each graph batch $B_i = (\mathbf{X}_i, \mathbf{A}_i)$ introduces new information. In inductive graphs, a batch contains newly added nodes and their edges. In transductive graphs, it typically contains newly revealed labels for existing nodes while the overall structure is retained [32]. These batches are progressively integrated, constructing a series of snapshot graphs $\{G_1, G_2, \ldots, G_m\}$. At any time step t, the snapshot graph $G_t = (\mathbf{X}_t, \mathbf{A}_t)$ encompasses all information up to that point, where $G_t = \bigcup_{i=1}^{t} B_i$. Given the snapshot graph G_t at the current time step t and the condensed graph G'_{t-1} from the previous step, the goal of evolving graph condensation is to find an update function $\mathcal{F}$ that generates a new condensed graph $G'_t = \mathcal{F}(G'_{t-1}, B_t)$. The objective is for G'_t to be significantly smaller than G_t while preserving essential structural and feature information, such that any GNN trained on G'_t achieves performance on downstream tasks that is comparable to a GNN trained on the full, original graph G_t. At each time step t, instead of repeating the GC process from scratch on G_t, our method aims to efficiently update the condensed graph. It effectively inherits knowledge from the previously condensed graph G'_{t-1} and integrates the new information from batch B_t to produce G'_t. This updated graph is then used to train GNNs, which are deployed to classify nodes in the full graph G_t. This incremental approach ensures that the condensed representation stays current with the evolving graph while minimizing redundant computation.

4 Methodology

In this section, we outline our condensation objectives for preserving training data information and generalizing to test data in static graphs. We employ theoretical analysis to simplify these objectives and introduce a scalable condensation method that does not require training GNN models, while offering traceability by establishing clear correspondence between the original and condensed nodes. Building on this scalable approach, we further propose an efficient adaptation to the evolving scenarios.

4.1 A Deep Dive into Condensation Objectives

Intuitively, the condensation process should preserve sufficient information from training data to maintain GNN training performance while ensuring model generalization to test data. Thus, we divide our discussion into two stages: *training* and *test*. To simplify our analysis, we adopt the Simplified Graph Convolution (SGC) model as the GNN [36], due to its simpler design, similar filtering behavior to GCN [20], and its frequent use as a backbone and evaluation model in numerous graph condensation works [10, 18, 19, 38].

Training Stage Objectives. During the training stage, GC aims to preserve the training data information to maintain the performance of GNNs. To reflect this, a natural way is to match the model predictions on the original graph G and its condensed counterpart G'. Denote the predictions of an SGC model trained on G as $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times c}$ and those on G' as $\hat{\mathbf{Y}}' \in \mathbb{R}^{N' \times c}$. We have $\hat{\mathbf{Y}} = \mathbf{FW} =$

KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

Shengbo Gong, Mohammad Hashemi, Juntong Ni, Carl Yang, and Wei Jin

$\mathbf{A}^K \mathbf{XW}$ and $\hat{\mathbf{Y}}' = \mathbf{F}'\mathbf{W}' = \mathbf{A}'^K \mathbf{X}'\mathbf{W}' \in \mathbb{R}^{N' \times d}$, where K denotes the number of SGC layers, $\mathbf{F}$ and $\mathbf{F}'$ represent the propagated feature matrices on G and G', and $\mathbf{W}$ and $\mathbf{W}'$ are the weight matrices of the SGC model trained on G and G', respectively. Note that feature propagation here is more flexible than the SGC model and is not limited to the formulation $\mathbf{A}^K \mathbf{X}$; rather, it can be expressed as any linear combination of powers of the adjacency matrix [29]. Below, we refer to the propagated feature as the node *representations*. We first define the distance between two matrices as the L2 norm of their difference. If matrices do not have the same shape, a projection matrix can be applied to transform them into a common shape before computing the distance. Then, the following theorem holds:

THEOREM 4.1. *The prediction distance in the training stage is bounded by the sum of representation distance and parameter distance.*

$$\|\mathcal{K}(\hat{\mathbf{Y}}) - \hat{\mathbf{Y}}'\| \le \|\mathcal{K}(\mathbf{F}) - \mathbf{F}'\| \cdot \|\mathbf{W}'\| + \|\mathbf{F}\| \cdot \|\mathbf{W} - \mathbf{W}'\| \quad (1)$$

where $\| \cdot \|$ denotes the L2 norm and $\mathcal{K}(\cdot)$ can be any projection function that aligns the dimensions of $\hat{\mathbf{Y}}$ and $\hat{\mathbf{Y}}'$ or $\mathbf{F}$ and $\mathbf{F}'$.

We provide proof in Appendix A.1. Note that $\|\mathbf{F}\|$ is a constant, and the weight matrix $\|\mathbf{W}'\|$ is naturally constrained due to regularization techniques during model optimization to control its magnitude. Therefore, Theorem 4.1 indicates that by minimizing the representation distance ($\|\mathcal{K}(\mathbf{F}) - \mathbf{F}'\|$) and parameter distance ($\|\mathbf{W} - \mathbf{W}'\|$) between two graphs, the predictions derived from the condensed graph can be close to those of the original graph.

Test Stage Objectives. At the test stage, our condensation goal is to ensure that the GNN model, trained on condensed training data G', generalizes effectively to the test graph, i.e., achieving a low prediction error on the test data. We slightly abuse the notation of $\mathbf{F}$ to represent the propagated feature matrix in the test graph, and denote the test ground truth label as $\mathbf{Y}$ and the predicted test labels from the model trained on G' as $\hat{\mathbf{Y}}'' = \mathbf{FW}'$. Then we have the following theorem that provides understanding for the test prediction error:

THEOREM 4.2. *The test prediction error of the GNN trained of G' is bounded by the test prediction error of the GNN trained on G plus the parameter distance, as formularized by $\|\mathbf{Y} - \hat{\mathbf{Y}}''\| \le \|\mathbf{Y} - \mathbf{FW}\| + \|\mathbf{F}\| \cdot \|\mathbf{W} - \mathbf{W}'\|$.*

We provide proof in Appendix A.2. This inequality incorporates both the original test prediction error and the parameter distance. It indicates that by reducing the parameter distance $\|\mathbf{W} - \mathbf{W}'\|$, the test prediction error becomes more tightly bounded, assuming that the original test prediction error $\|\mathbf{Y} - \mathbf{FW}\|$ and the propagated feature matrix $\mathbf{F}$ remain constant.

Summary - Reframing Objectives for Graph Condensation. Our analysis leads us to a new approach to defining the objectives for GC. Theorems 4.1 and 4.2 suggest that both training and test stage objectives are upper bounded by the parameter distance $\|\mathbf{W} - \mathbf{W}'\|$. Furthermore, the training stage objectives are additionally upper bounded by the representation distance $dist(\mathbf{F}, \mathbf{F}')$. Thus, we introduce a new GC objective that focuses on minimizing both the **parameter and representation distances** to optimize the overall performance of the condensed graphs.

4.2 Efficient Optimization for the New Objective

Focusing on the new condensation objective we propose, we now discuss its efficient optimization and the establishment of a correspondence between original and condensed nodes to enhance traceability. As our objective includes representation and parameter distance, we separate their discussions as follows.

Optimizing Representation Distance $\|\mathcal{K}(\hat{\mathbf{Y}}) - \hat{\mathbf{Y}}'\|$. As discussed in Theorem 1, $\mathcal{K}(\cdot)$ serves as a projection function that aligns the dimensions of $\hat{\mathbf{Y}}$ and $\hat{\mathbf{Y}}'$. While there are multiple choices of $\mathcal{K}(\cdot)$, we propose to implement it through a linear mapping that assigns each node in the original graph G to one of the synthetic nodes in G'. This mapping is formalized as a function $\pi : \{1, \ldots, N\} \rightarrow \{1, \ldots N'\}$, represented by the assignment matrix $\mathbf{P} \in \mathbb{R}^{N \times N'}$. The matrix $\mathbf{P}$ is binary, where $\mathbf{P}_{ij} = 1$ if and only if node i in G is assigned to node j in G', i.e., $\pi(i) = j$. To align $\hat{\mathbf{Y}}$ with $\hat{\mathbf{Y}}'$, we first transform $\hat{\mathbf{Y}}$ to the corresponding dimension using $\mathbf{P}^\top \hat{\mathbf{Y}}$. We then compute the prediction distance as $\|\mathbf{P}^\top \hat{\mathbf{Y}} - \hat{\mathbf{Y}}'\|$, which modifies the minimization of representation distance as follows:

$$\min_{\mathbf{F}'} \|\mathcal{K}(\mathbf{F}) - \mathbf{F}'\| = \min_{\mathbf{P}, \mathbf{F}'} \|\mathbf{P}^\top \mathbf{F} - \mathbf{F}'\|. \quad (2)$$

By solving the above optimization problem, we can derive the assignment matrix $\mathbf{P}$ and the propagated features $\mathbf{F}'$ for graph G'. In alignment with the popular structure-free GC paradigm [19, 41, 42], we set $\mathbf{F}'$ as the condensed node features and use an identity matrix $\mathbf{I}$ for the condensed adjacency matrix to form a condensed graph that effectively minimizes the representation distance. Notably, the assignment matrix $\mathbf{P}$ and the condensed features $\mathbf{F}'$ can be efficiently derived using any Expectation-Maximization (EM)-based clustering algorithm, such as k-means. Since k-means inherently minimizes the Sum of Squared Errors (SSE), this optimization aligns with the objective $\min \|\mathbf{PC} - \mathbf{F}\|$, where $\mathbf{C}$ represents the cluster centers of feature points $\mathbf{F}$. The procedure iteratively updates the cluster centroids $\mathbf{F}'$ and the assignment matrix $\mathbf{P}$ until convergence is reached. This approach presents two major advantages: (a) **Traceability**: It provides clear insight into how each original node contributes to the condensed graph, facilitating a better understanding of the condensation process. With traceability, users can manipulate the condensed graph alongside the original one—for example, easily filtering low-quality data when the corresponding original nodes exhibit poor characteristics. (b) **Efficiency and Versatility**: Unlike traditional GC methods that need specific GNN models to be trained, this new method does not require any GNN training or gradient calculation and is not limited to a particular GNN architecture, which greatly improves efficiency.

Optimizing Parameter Distance $\|\mathbf{W} - \mathbf{W}'\|$. As demonstrated in the above establishment of structure-free condensed graphs, clustering effectively bridges the gap between original and condensed graphs by minimizing representation distance. Within this framework, we further explore how the parameter distance correlates with the assignment matrix $\mathbf{P}$ used in clustering through the following theorem.

THEOREM 4.3. *The parameter distance can be bounded by the following inequality:*

$$\|\mathbf{W} - \mathbf{W}'\| \le C \cdot (\max(diag(\mathbf{P}^\top \mathbf{P})))^2, \quad (3)$$

Scalable Graph Condensation with Evolving Capabilities

KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

where $C = \frac{\|\mathbf{F}\| \cdot \|\mathbf{Y}\| (\lambda_{\min}(\mathbf{F}^\top \mathbf{F}) + \|\mathbf{F}\|)}{\lambda_{\min}^2(\mathbf{F}^\top \mathbf{F})}$ is a constant, $\mathbf{P}^\top \mathbf{P} \in \mathbb{R}^{N' \times N'}$ is a diagonal matrix with each diagonal entry corresponding to how many original nodes are assigned to each synthetic node.

We provide proof of Theorem 4.3 in Appendix A.3. This theorem demonstrates that the parameter distance $\|\mathbf{W} - \mathbf{W}'\|$ is upper bounded by the maximum elements in $\mathbf{P}^\top \mathbf{P}$. It suggests that assigning a balanced number of original nodes to each synthetic node can effectively lower the upper bound of the parameter distance. Thus, we plan to integrate this balancing constraint into our condensation process to facilitate the minimization of parameter distance.

4.3 Graph Condensation via Clustering

With clustering established as a powerful and efficient approach for achieving bounded error in GC, we now shift our focus to its practical implementation. To facilitate a clustering-based GC method, we propose the GECC process. We start with the foundational GC paradigm, which provides an efficient method for generating the condensed graph G'_t from the static original graph G_t at time step t. Building on this, we extend to an evolving graph setting by introducing the incremental condensation paradigm to handle dynamic graph updates efficiently. GECC takes the original graph at time step t and the condensed node representations from the previous time step $t-1$ as input, generating the condensed node representations for the current time step. GECC process can be divided into two main stages: *(i.)* **Feature Propagation**, and *(ii.)* **Representation Clustering**. The details of these steps are explained below.

4.3.1 Feature Propagation. The goal of feature propagation is to propagate information from a node's neighbors to provide each node with a richer representation. To enable a GC procedure that eliminates the need for training on the whole graphs, we develop a non-parametric feature propagation module [12, 27, 28, 36] designed to produce node embeddings $\mathbf{F}_t$ in the original graph G_t. In our work, by following the propagation method in SGC [36], the propagated node representations after k steps are computed as: $\mathbf{F}_{t,k} = \hat{\mathbf{A}}_t^k \mathbf{X}_t$, where $\hat{\mathbf{A}}_t$ is the normalized adjacency matrix of G_t, defined as: $\hat{\mathbf{A}}_t = \tilde{\mathbf{D}}_t^{-\frac{1}{2}} \tilde{\mathbf{A}}_t \tilde{\mathbf{D}}_t^{-\frac{1}{2}}$, with $\tilde{\mathbf{A}}_t = \mathbf{A}_t + \mathbf{I}$ being the adjacency matrix with self-loops, and $\tilde{\mathbf{D}}_t$ being the degree matrix of $\tilde{\mathbf{A}}_t$, where $(\tilde{\mathbf{D}}_t)_{i,i} = \sum_j \tilde{\mathbf{A}}_{t,i,j}$. To combine information from different propagation steps, we adopt a linearization approach. The final representation for each node is given by: $\mathbf{F}_t = \sum_{k=0}^{K} \alpha_k \mathbf{F}_{t,k}$. This formulation ensures that each node's representation captures multi-hop neighborhood information, which fuse the structure and feature information. Note that we allow the weights α to vary in a large range. In particular, if $\alpha < 0$, it effectively introduces a "negative offset" that can capture heterophilic relationships in the representation [43].

4.3.2 Representation Clustering. Once the node representations $\mathbf{F}_t \in \mathbb{R}^{n_t \times d}$ are obtained, we discard the original graph structure and partition the node representations into groups via the following clustering technique. Each node in the condensed graph G'_t belongs to one of c classes. We denote their labels as $\mathbf{y}'_t \in \{1, 2, \ldots, c\}^{n'_t}$ or equivalently via one-hot encoding $\mathbf{Y}'_t \in \mathbb{R}^{n'_t \times c}$. Following the approach of GCond [19], we predefine the labels of the condensed

nodes so that their class distribution in $\mathbf{y}'_t$ matches the original distribution in $\mathbf{y}_t$.

A Unified Clustering Formulation. By leveraging the node labels and their feature representations $\{\mathbf{F}_{t,k} \mid k = 1, \ldots, c\}$, we can apply a variety of clustering methods to partition the nodes. Here, we adopt a unified formulation that accommodates both *hard k*-means and *soft k*-means. We first split the labeled nodes by their class label $k \in \{1, 2, \ldots, c\}$. Within class k, we assign the nodes to M_k clusters, where M_k is chosen according to a desired *reduction rate* r[1]. Let $M = \sum_{k=1}^{c} M_k = n_{t,k} \times r$, where $n_{t,k}$ is number of training nodes in class k. Next, we formulate the (hard/soft) assignment matrix $\mathbf{P}_t \in \mathbb{R}^{n_t \times M}$. Each row corresponds to one labeled node, and each column corresponds to M clusters across all classes. In more fine-grained notation, if cluster j belongs to class k, then the entry $\mathbf{P}_{t,i,j}$ indicates how node i of class k is associated with that cluster. Formally,

$$\mathbf{P}_{t,i,j} \in \begin{cases} \{0, 1\}, \text{ hard clustering.} \\ [0, 1], \text{ soft clustering: subject to } \sum_j \mathbf{P}_{t,i,j} = 1. \end{cases} \tag{4}$$

In the hard-assignment scenario, each node belongs to exactly one cluster, so $\mathbf{P}_{t,i,j} = 1$ if node i is placed in cluster j and 0 otherwise. In the soft-assignment scenario, each node has fractional memberships across the clusters in its class, with the memberships summing to 1.

Under hard k-means, each node belongs to exactly one cluster. Under soft k-means, each node has fractional memberships that sum to 1. We employ the classic algorithms of k-means [25] and fuzzy c-means [2] in practice.

A Balanced SSE Objective. As Theorem 4.3 indicates, assigning a balanced number of original nodes to each synthetic node can effectively reduce the upper bound of the parameter distance, thereby enhancing the objectives in both the training and testing stages. Thus, to obtain balanced clusters, we add a regularization term that penalizes deviations from a uniform cluster size. Concretely, let $\mathbf{C}_t \in \mathbb{R}^{M \times d}$ collect the centroids for the M total clusters across all classes. We denote $\mathbf{1} \in \mathbb{R}^{n_t}$ as the all-ones vector, where n_t is the total number of labeled nodes. Suppose class k has $n_{t,k}$ labeled nodes and is assigned M_k clusters, so that $M = \sum_{k=1}^{c} M_k$. To encourage each class to be evenly partitioned among its M_k clusters, we set

$$\mathbf{u} = \Big[\underbrace{\frac{n_{t,1}}{M_1}, \ldots, \frac{n_{t,1}}{M_1}}_{M_1 \text{ entries}}, \underbrace{\frac{n_{t,2}}{M_2}, \ldots, \frac{n_{t,2}}{M_2}}_{M_2 \text{ entries}}, \ldots, \underbrace{\frac{n_{t,c}}{M_c}, \ldots, \frac{n_{t,c}}{M_c}}_{M_c \text{ entries}} \Big]^\top,$$

so that each block of $\mathbf{u}$ is constant, corresponding to the perfectly balanced cluster size for each class. Since clustering can be stochastic, we repeat it multiple times and choose the best result. The standard metric for clustering quality is the sum of squared errors (SSE). Finally, the balanced SSE objective is then given by:

$$J(\mathbf{P}_t, \mathbf{C}_t) = \left\| \mathbf{F}_t - \mathbf{P}_t \mathbf{C}_t \right\|^2 + \left\| \mathbf{P}_t^\top \mathbf{1} - \mathbf{u} \right\|^2. \tag{5}$$

Centroid Computation. Regardless of whether the assignment is hard or soft, the cluster centroids are computed by:

$$\mathbf{C}_t = \mathbf{D}_{P_t}^{-1} \mathbf{P}_t^\top \mathbf{F}_t, \tag{6}$$

where $\mathbf{D}_{P_t} \in \mathbb{R}^{n_t \times n_t}$ is a diagonal matrix satisfying $(\mathbf{D}_{P_t})_{k,k} = \sum_{i=1}^{n_t} \mathbf{P}_{t,i,k}$. Finally, as the propagated features implicitly encode

---

[1]Reduction rate r is defined as (#nodes in synthetic set)/(#nodes in training set).

KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

Shengbo Gong, Mohammad Hashemi, Juntong Ni, Carl Yang, and Wei Jin

structural information, and current SOTA methods [41, 42] operate on structure-free graphs, we do not explicitly generate new edges. Beyond achieving strong predictive performance, this structure-free design is significantly more efficient than structure-based approaches, which typically require an additional edge-generation process with $O((n_t r)^2)$ complexity [10].

Instead, we use an identity matrix to represent the adjacency among condensed nodes. By combining feature propagation with our representation-based clustering, we form the condensed graph as $G'_t = (C'_t, I'_t)$, where $I'_t \in \mathbb{R}^{M \times M}$ is an identity matrix. This condensed graph preserves the essential structural and feature information of the original graph G_t, thereby enabling efficient GNN training while substantially reducing computational overhead.

4.4 Evolving Condensation via Incremental Initialization

To enable the GECC pipeline to effectively condense graphs that evolve over time and increase in size, we adopt a clustering approach inspired by the K-means++ initialization technique [1]. This method improves the quality of clustering by ensuring a better spread of centroids during initialization. Specifically, for each node class c, we partition the nodes into C clusters using the following process:

First, let the set of already selected centroids at step $t - 1$ be $C_{t-1} = \{c_1, c_2, \ldots, c_k\}$, where $c_i \in \mathbb{R}^d$, and let the new coming node set in new time step is $\{x_1, x_2, \ldots, x_m\}$. For the current step t, we compute $D(x_i, C_{t-1})$ for each node embedding $x_i \in \mathbb{R}^d$ in class c to consider how well the current centroids represent the data points in the feature space and identify regions of the space that require additional centroids. The distance is defined as the minimum Euclidean distance between x_i and any centroid in C_{t-1}:

$$D(x_i, C_{t-1}) = \min_{c \in C_{t-1}} \|x_i - c\|^2.$$

Next, we probabilistically select a new centroid c_{k+1} from the set of node embeddings $\{x_i\}$ based on the computed distances. The probability of selecting x_i is proportional to $D(x_i, C_{t-1})^2$:

$$P(x_i) = \frac{D(x_i, C_{t-1})^2}{\sum_{x_j \in \{x_1, x_2, \ldots, x_m\}} D(x_j, C_{t-1})^2}.$$

This step ensures that new centroids are more likely to be chosen from areas of the feature space that are underrepresented, thus improving the spread of the centroids. In an evolving setting, we expect the condensed graph to grow proportionally with the original graph. Therefore, we sample $m \times r$ new centroid sets, denoted by $\{C_\Delta\}$. We then update the set of centroids as follows:

$$C_t = C_{t-1} \cup \{C_\Delta\}.$$

Finally, after obtaining C_t, each node embedding x_i in class c is assigned to the nearest centroid:

$$\text{Cluster}(x_i) = \arg\min_{c \in C_t} \|x_i - c\|^2.$$

This step partitions the nodes into C clusters, ensuring that each cluster is represented by its respective centroid.

Once the centroids C_t are obtained through this process, they will be served as the condensed node representations X'_t in the synthetic graph G'_t at time step t. By leveraging this probabilistic initialization, the clustering method ensures that new centroids

are well-distributed and far from existing ones, facilitating a more effective condensation of graph representations at each time step. The complexity analysis can be found in Appendix D.

5 Experiments

To validate the effectiveness of our proposed GECC, we compare it against classic and SOTA baselines in both non-evolving and evolving scenarios. We first detail the experimental setup, including the construction of benchmark datasets and the experimental settings of each method. Next, we present the node classification performance as a measure of condensation results, alongside a comparison of efficiency. Finally, we empirically demonstrate the effectiveness of the feature aggregation module and the importance of the specific incremental initialization design in our method. Our code is provided in https://github.com/rockcor/GECC.

5.1 Experimental Setup

Datasets and Baselines. Following most of the GC papers, we select seven datasets: five transductive datasets, i.e., Citeseer, Cora [20], Pubmed [30], Ogbn-arxiv, and Ogbn-products [16] and two inductive datasets, Flickr and Reddit [40]. All training graphs are randomly divided into five subsets and each preserving the original class distribution except for Ogbn-arxiv-real, which follows the real timestamps of the publication years. For transductive graphs, nodes in training set are split, whereas inductive graphs are partitioned into subgraphs. Subsequently, the training sets are **incrementally enlarged**—for example, the first subset forms G_1, and the first plus the second subset forms G_2. For additional dataset details, please refer to Appendix E.1. We compare GECC with most effective and efficient GC methods, encompassing a diverse range of optimization strategies: (1) gradient matching-based: GCond and GCondX [19]; (2) distribution matching-based: GCDM [22] and SimGC [38]; (3) trajectory matching-based: GEOM [41]. We also include node selection baselines (Random [19], KCenter [31], Herding [35]) to better compare performance–efficiency trade-offs. The condensed graphs are evaluated using a standard GCN trained for 300 epochs with a learning rate of 0.01, as suggested by GC4NC [10]. The GCN is trained on the synthetic dataset and validated and tested on the original validation and test sets. We also list the results from training a standard GCN in whole dataset for both two settings to show the potential upper bound for graph condensation.

Implementation Details. To ensure a fair reproduction and comparison of baseline methods, we use the best hyperparameters reported in their original papers. For intermediate evaluation, we follow the GC4NC benchmark [10], which restricts the number of evaluations to 10 during the whole condensation process. All baselines adopt the training from scratch strategy in evolving setting, i.e., do not reuse the previous condensed graphs. For the hyperparameters of our method, we tune them within a limited range, specifically $\alpha_0, \alpha_1, \alpha_2 \in [-0.3, 0.9]$ with 0.1 interval. We introduce a negative offset to capture heterophilious properties in graphs [11, 43]. Following prior work, we fix the maximum propagation depth in Equation 4.3.1 as $K = 2$. In addition, learning rate, epochs and dropout of downstream GCN are all fixed as 0.01, 300 and 0.5, except the weight decay is selected from $\{0.001, 0.0005\}$. We use soft clustering for small datasets and the repeat times are set

Table 2: Comparison of different condensation methods in two settings and seven datasets. The evolving setting row show the average test accuracy of five time steps. The best results are in bold and the second-best are <u>underlined</u>. "OOM" indicates out-of-memory errors. Average condensation time (seconds) for both settings are listed alongside test accuracy for each method.

| Dataset	Setting	Random	Herding	KCenter	GCondX		GCDM		GCond		SimGC		GEOM		GECC		Whole
		Acc. ↑	Acc. ↑	Acc. ↑	Acc. ↑	Time ↓	Acc. ↑	Time ↓	Acc. ↑	Time ↓	Acc. ↑	Time ↓	Acc. ↑	Time ↓	Acc. ↑	Time ↓	Acc. ↑
Citeseer	Non-Evolving	62.62	66.66	59.04	68.38	506	69.35	654	72.08	218	66.40	1680	<u>73.03</u>	1362	**73.25**	*1.7*	72.11
	Evolving	50.65	53.47	47.99	50.85		60.51		<u>61.51</u>		57.42		58.95		**65.48**		63.57
Cora	Non-Evolving	72.24	73.77	70.55	78.60	332	80.54	1190	80.68	143	79.60	1644	<u>82.82</u>	1331	**82.99**	*1.7*	81.23
	Evolving	58.00	63.07	59.90	67.18		<u>77.14</u>		74.54		64.42		72.56		**77.36**		76.34
Pubmed	Non-Evolving	71.84	75.53	74.00	71.97	247	76.46	502	77.48	311	76.80	1654	<u>78.49</u>	995	**80.24**	*1.4*	78.65
	Evolving	66.37	66.31	64.38	62.65		74.26		<u>74.49</u>		71.38		70.25		**76.74**		76.18
Flickr	Non-Evolving	44.68	45.12	43.53	46.58	610	**46.99**	1447	45.88	354	41.01	7487	46.13	758	<u>46.63</u>	*7.1*	47.53
	Evolving	44.70	44.66	44.33	<u>45.63</u>		45.52		44.98		41.94		45.43		**45.78**		46.97
Ogbn-arxiv	Non-Evolving	60.19	57.70	58.66	59.93	2895	64.23	3020	60.71	686	65.26	2687	**69.59**	1685	<u>66.71</u>	*10*	70.95
	Evolving	56.04	57.57	56.21	60.73		62.50		59.98		64.97		**66.30**		65.42		70.40
Ogbn-arxiv-real	Non-Evolving	60.19	57.70	58.66	59.93	1328	64.23	1862	60.71	617	65.26	3240	**69.59**	1646	<u>66.71</u>	*8.6*	70.95
	Evolving	56.04	57.57	56.21	59.33		61.36		59.46		59.34		<u>58.28</u>		**61.48**		70.40
Ogbn-products	Non-Evolving	60.19	57.70	58.66	OOM	-	OOM	-	OOM	-	<u>61.71</u>	71489	OOM	-	**66.32**	*147*	73.40
	Evolving	41.36	44.26	38.93	OOM		OOM		OOM		<u>61.93</u>		OOM		**64.03**		73.88
Reddit	Non-Evolving	55.73	59.34	48.28	88.25	2673	89.82	6130	89.96	337	90.78	6611	<u>91.33</u>	1816	**91.37**	*4.9*	93.70
	Evolving	51.31	48.94	48.53	79.02		87.93		82.68		67.91		<u>89.85</u>		**90.02**		93.92
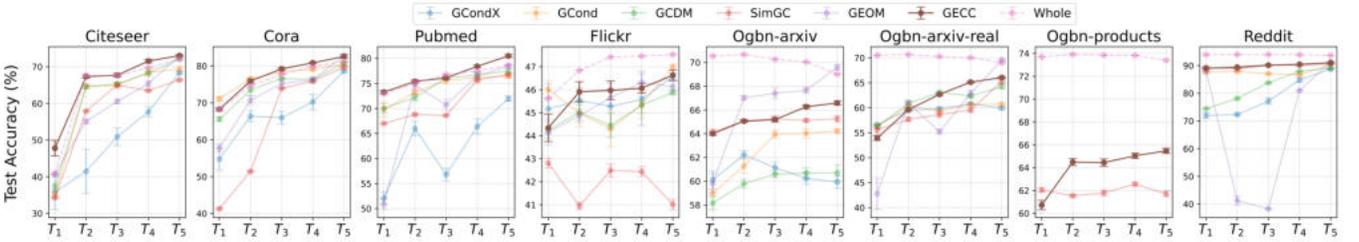
Figure 1: Comparison of test accuracy of different GC methods across five time steps.

to 50. The fuzziness are selected from {1.0, 1.1, 1.3}, respectively. For large datasets, we run standard hard *k*-means only one time. The maximum number of iterations and the *k*-means threshold are set to 300 and 1×10^{-8}, respectively. All experiments are run ten times then we report the average. See more details in Appendix E.

5.2 Performance and Efficiency Comparison in the non-Evolving and Evolving Setting

5.2.1 Performance Comparison. To compare the effectiveness of GECC with the baselines, we utilize the generated condensed graph data to train a standard GCN, reporting both test accuracies and standard deviations in Table 2. For each dataset, we report two test accuracies: one for the *Non-Evolving* setting, reflecting the accuracy achieved by training the GCN on the graph at the final time step ($t = 5$), corresponding to the largest graph size; and one for the *Evolving* setting, representing the average accuracy across all five time steps as the graph evolves. It is also noteworthy that, to evaluate our proposed method on real-world dynamic datasets such as *Ogbn-Arxiv-real*, we utilize the original metadata from *Ogbn-arxiv* [16] and use node timestamps to divide the data into five sequential time steps.

Additionally, Figure 1 illustrates the test accuracy across different time steps, which compares the GECC evolving capability with the existing baselines. Our analysis reveals several key insights:

Non-evolving setting – GECC outperforms the baselines on almost all datasets, including the *whole* dataset performance for some datasets, which highlights the effectiveness of our training-free approach in the non-evolving setting. The only exceptions are observed in the *Flickr* and *Ogbn-arxiv* datasets, where GECC ranks second by a minimal margin. In addition, unlike other baselines, GECC does not fail on extremely large datasets like *Ogbn-products* due to memory constraints.

Evolving setting – By analyzing Table 2, we observe that GECC consistently outperforms existing baselines across almost all datasets, often by a large margin. The only exception is *Ogbn-arxiv*, where GECC ranks second. ALso, Figure 1 highlights that GECC achieves the best test accuracy at early time steps. For instance, on *Ogbn-arxiv*, it surpasses 65% accuracy by the second time step, a large gap compared to existing baselines. This demonstrates GECC's efficiency in leveraging limited data for superior generalization, whereas other methods struggle to reach comparable performance early because they need sufficient data for training. For the *Ogbn-arxiv-real* dataset, a real-world graph with inherent dynamics based on original timestamps, GECC still achieves the best evolving performance, indicating that our method remains effective in realistic

Table 3: Transferability: GECC is compatible with various GNN architectures. "Average" denotes the mean ± standard deviation of test accuracy across different GNN models.

Model	Cora ($r = 0.5$)			Ogbn-arxiv ($r = 0.01$)			Reddit ($r = 0.001$)		
	GCond	GEOM	GECC	GCond	GEOM	GECC	GCond	GEOM	GECC
GCN	81.69	83.06	82.71	65.83	69.49	66.60	88.01	91.57	91.28
SGC	81.38	83.15	82.67	64.76	67.51	65.55	88.69	90.58	92.10
APPNP	80.80	83.33	81.97	64.97	67.03	64.45	85.90	89.31	91.07
Cheby	77.61	80.49	78.45	60.52	62.35	60.17	74.51	82.58	82.26
GraphSage	77.52	74.12	78.45	60.38	62.19	60.24	74.47	82.60	82.24
GAT	76.47	81.85	82.09	61.90	64.70	64.35	88.84	89.19	90.40
Average	79.25±1.76	<u>81.00±3.09</u>	**81.06±2.01**	63.06±2.23	**65.55±2.74**	<u>63.56±2.31</u>	83.40±5.41	<u>87.64±3.77</u>	**88.23±3.28**

dynamic scenarios. **Moreover**, we observe that GECC surpasses the performance achieved on the whole dataset for relatively smaller graphs such as *Cora*, *Citeseer*, and *Pubmed*. This result highlights the effectiveness of our training-free condensation approach in achieving a "lossless" objective during graph evolution. **Additionally**, Figure 1 illustrates the robustness of GECC in steadily improving performance as training size increases. This trend fails to hold for baselines, particularly for GEOM on *Reddit*. We conjecture that the trajectory matching method heavily relies on the performance of the pre-trained GNN, making it susceptible to the twofold influence of data size changes: first, during the pre-training stage, and second, during the condensation stage.

5.2.2 Efficiency Comparison. Comparing GECC with the baselines in Figure 2, it is evident that GECC exhibits superior efficiency and scalability. It maintains stable performance while effectively managing computational resources as the graph evolves. Numerical experiments show sublinear runtime growth: a fitted power-law exponent (≈ 0.3) indicates reasonably good scalability. Although certain model-based GC methods such as GEOM may slightly outperform GECC at specific time steps, GECC achieves over 100 times faster condensation time and demonstrates a significantly slower increase in computational overhead. For results in more datasets, please refer to Appendix E.
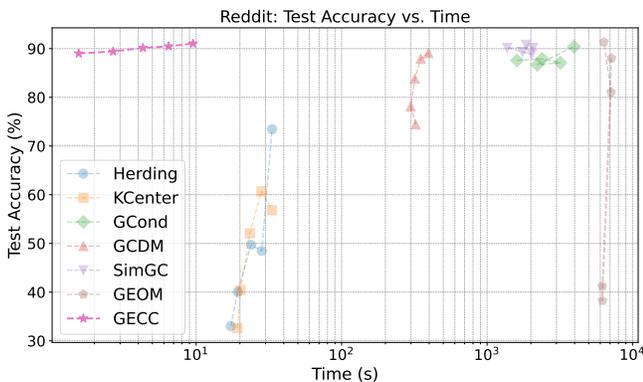
Figure 2: Test accuracy vs. condensation time on the *Reddit* dataset (top-left is better).

5.2.3 Transferability. A crucial factor in evaluating GC methods is determining whether the condensed data can effectively train various GNNs from a data-centric perspective. Unlike GECC, which adopts a training-free condensation approach, most existing methods generate condensed graphs that are inherently dependent on the backbone GNN used during condensation, such as GCN [10, 14]. This reliance can introduce inductive biases, potentially hindering their adaptability to other GNN architectures.

Table 3 shows that the condensed graphs generated by GECC demonstrate robust generalization across diverse architectures when compared to other SOTA baselines. Even in **Ogbn-arxiv**, GECC outperforms GEOM w.r.t consistency, as indicated by its lower standard deviation across downstream GNN models, highlighting the advantage of GECC's model-agnostic design.

5.3 Ablation Studies

To investigate the effectiveness of each module of our method, we conduct the following ablation studies to study the impact of feature propagation, incremental k-means++, and balanced SSE score.

5.3.1 Impact of Feature Propagation. As feature propagation (Equation 4.3.1) is crucial for generating informative node representations, we compare GECC against an ablated version without feature propagation (labeled *w/o propagation*) across all time steps. Specifically, for *w/o propagation*, we set $\alpha_0 = 1$ and all other α coefficients to 0, keeping all other hyperparameters identical. Each experiment is repeated ten times, and we report the average results.

Figure 3 illustrates that GECC with feature propagation outperforms the version without propagation. Notably, in some datasets (e.g., Flickr), the *w/o propagation* approach exhibits a downward trend even as more graph data is introduced, suggesting that the raw node features may contain substantial noise. In contrast, GECC with feature propagation steadily improves as the dataset size increases, highlighting that propagated features effectively mitigate noise and bolster clustering performance.

5.3.2 Impact of Incremental k-Means++. We evaluate the effect of reusing previous condensed results when clustering at each time step. Specifically, we compare GECC against a variant that does *not* reuse prior centroids—denoted *w/o incremental k-means++*—and instead initializes centroids from scratch via a standard k-means++ procedure at every time step.

Scalable Graph Condensation with Evolving Capabilities

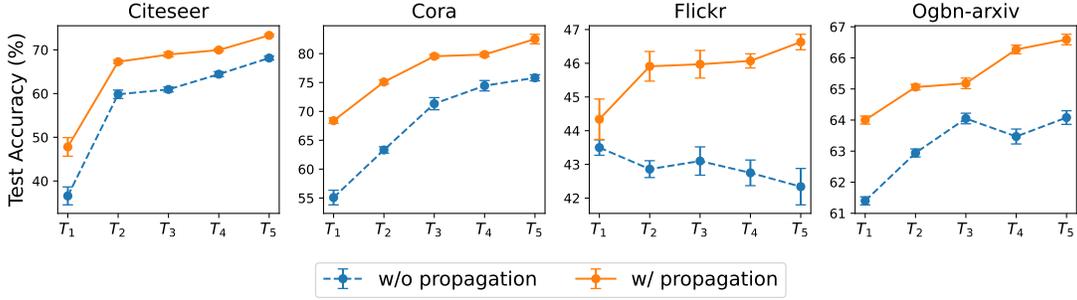KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

Figure 3: Performance comparison between GECC and GECC without feature aggregation.
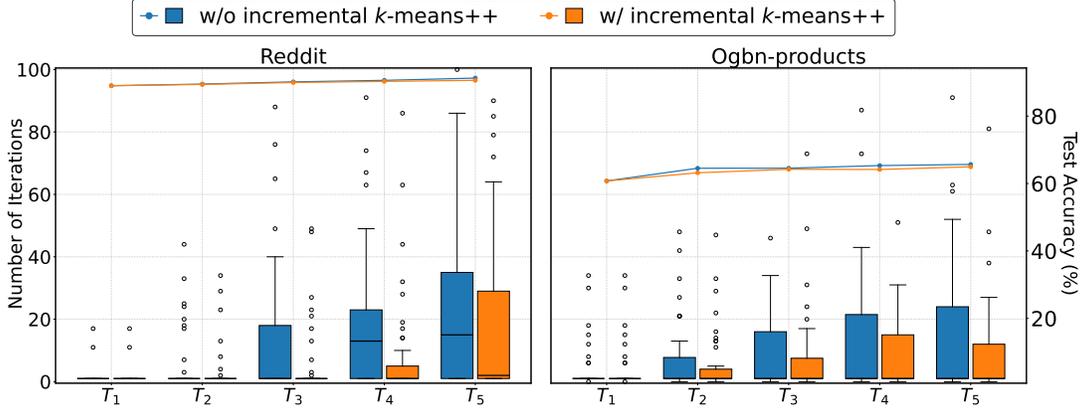

Figure 4: Ablation study. Boxplots show the mean and quartiles of the number of iterations required for clustering.

Table 4: Comparison of average test accuracy (%) and balanced SSE on three benchmark datasets across five time steps. "Bal. SSE" indicates the balanced SSE.

	Citeseer		Cora		Pubmed	
	Acc. (↑)	Bal. SSE (↓)	Acc. (↑)	Bal. SSE (↓)	Acc. (↑)	Bal. SSE (↓)
w/ rep. clustering	65.45	2.39	77.08	4.52	76.32	6.65
w/o rep. clustering	63.75	9.83	74.82	9.47	75.99	9.89

As shown in Figure 4, reusing previously learned cluster centers via incremental k-means++ significantly reduces the required number of iterations for convergence, especially as the graph grows larger. For example, on the *Reddit* dataset at time step 5, incremental initialization only requires about 10% of the iterations needed when initializing from scratch. We omit results on smaller datasets because they typically converge in fewer than 10 iterations.

5.3.3 Relation between GC and clustering objective. The balanced Sum of Squared Errors (SSE) is a key contribution derived from our theoretical analysis. To assess its impact, we perform an ablation study comparing the performance of GECC with and without repetitive clustering. As shown in Table 4, for instance, on the *Citeseer* dataset, applying repetitive clustering to select the lowest SSE leads to an absolute performance improvement of 2.7%. These results show that a lower SSE correlates with higher test accuracy.

6 Conclusion and Outlook

In this study, we address the challenge of evolving graph condensation. We observe that a universal clustering framework can naturally optimize the assignment matrix, thereby achieving the common objectives of existing GC methods. Additionally, we propose a novel *balanced SSE* metric that further tightens the upper bound of these objectives. In the evolving setting, we find that our clustering approach can be readily adapted to an incremental version, termed *incremental k-means++*.

Experimental results demonstrate that balanced SSE improves the performance of clustering-based GC, and incremental k-means++ significantly reduces the number of iterations, thereby enhancing efficiency in evolving environments. Future work includes developing more efficient and scalable clustering techniques, especially soft clustering algorithms for larger graph datasets and adaptively optimizing multi-hop weights, which could be beneficial when the graph keeps evolving over time.

7 Acknowledgement

KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

Shengbo Gong, Mohammad Hashemi, Juntong Ni, Carl Yang, and Wei Jin

References

[1] David Arthur and Sergei Vassilvitskii. 2006. *k-means++: The advantages of careful seeding.* Technical Report. Stanford.

[2] James C Bezdek, Robert Ehrlich, and William Full. 1984. FCM: The fuzzy c-means clustering algorithm. *Computers & geosciences* 10, 2-3 (1984), 191–203.

[3] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

[4] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. 2019. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems* 32 (2019).

[5] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference.* 417–426.

[6] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds.*

[7] Xinyi Gao, Tong Chen, Yilong Zang, Wentao Zhang, Quoc Viet Hung Nguyen, Kai Zheng, and Hongzhi Yin. 2024. Graph condensation for inductive node representation learning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE).* IEEE, 3056–3069.

[8] Xinyi Gao, Tong Chen, Wentao Zhang, Yayong Li, Xiangguo Sun, and Hongzhi Yin. 2024. Graph condensation for open-world graph learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 851–862.

[9] Xinyi Gao, Tong Chen, Wentao Zhang, Junliang Yu, Guanhua Ye, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2024. Rethinking and Accelerating Graph Condensation: A Training-Free Approach with Class Partition. *arXiv preprint arXiv:2405.13707* (2024).

[10] Shengbo Gong, Juntong Ni, Noveen Sachdeva, Carl Yang, and Wei Jin. 2024. GC4NC: A Benchmark Framework for Graph Condensation on Node Classification with New Insights. *arXiv preprint arXiv:2406.16715* (2024).

[11] Shengbo Gong, Jiajun Zhou, Chenxuan Xie, and Qi Xuan. 2023. Neighborhood homophily-based graph convolutional network. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management.* 3908–3912.

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[13] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. doi:10.1038/s41586-020-2649-2

[14] Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B Aditya Prakash, and Wei Jin. 2024. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. *IJCAI* (2024).

[15] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. 2021. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430* (2021).

[16] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[17] Kezhao Huang, Jidong Zhai, Zhen Zheng, Youngmin Yi, and Xipeng Shen. 2021. Understanding and bridging the gaps in current GNN performance optimizations. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.* 119–132.

[18] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. 2022. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 720–730.

[19] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. Graph condensation for graph neural networks. *arXiv preprint arXiv:2110.07580* (2021).

[20] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[21] Yuan Li, Jun Hu, Zemin Liu, Bryan Hooi, Jia Chen, and Bingsheng He. 2025. A Precompute-Then-Adapt Approach for Efficient Graph Condensation. https://openreview.net/forum?id=kwagvI8Anf

[22] Mengyang Liu, Shanchuan Li, Xinshi Chen, and Le Song. 2022. Graph condensation via receptive field distribution matching. *arXiv preprint arXiv:2206.13697* (2022).

[23] Yilun Liu, Ruihong Qiu, and Zi Huang. 2023. Cat: Balanced continual graph learning with graph condensation. In *2023 IEEE International Conference on Data Mining (ICDM).* IEEE, 1157–1162.

[24] Zewen Liu, Guancheng Wan, B Aditya Prakash, Max SY Lau, and Wei Jin. 2024. A review of graph neural networks in epidemic modeling. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 6577–6587.

[25] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[26] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. 2024. Classic gnns are strong baselines: Reassessing gnns for node classification. *Advances in Neural Information Processing Systems* 37 (2024), 97650–97669.

[27] Nan Ma, Jiancheng Guan, and Yi Zhao. 2008. Bringing PageRank to the citation analysis. *Information Processing & Management* 44, 2 (2008), 800–810.

[28] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. 2021. Improving graph neural networks with simple architecture design. *arXiv preprint arXiv:2105.07634* (2021).

[29] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. 2022. Simplifying approach to node classification in graph neural networks. *Journal of Computational Science* 62 (2022), 101695.

[30] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. 2012. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs,* Vol. 8. 1.

[31] Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489* (2017).

[32] Junwei Su, Difan Zou, Zijun Zhang, and Chuan Wu. 2023. Towards robust graph incremental learning on evolving graphs. In *International Conference on Machine Learning.* PMLR, 32728–32748.

[33] Qingyun Sun, Ziying Chen, Beining Yang, Cheng Ji, Xingcheng Fu, Sheng Zhou, Hao Peng, Jianxin Li, and Philip S Yu. 2024. Gc-bench: An open and unified benchmark for graph condensation. *arXiv preprint arXiv:2407.00615* (2024).

[34] Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, and Qing Li. 2024. Fast graph condensation with structure-based neural tangent kernel. In *Proceedings of the ACM on Web Conference 2024.* 4439–4448.

[35] Max Welling. 2009. Herding dynamical weights to learn. In *Proceedings of the 26th annual international conference on machine learning.* 1121–1128.

[36] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning.* PMLR, 6861–6871.

[37] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[38] Zhenbang Xiao, Yu Wang, Shunyu Liu, Huiqiong Wang, Mingli Song, and Tongya Zheng. 2024. Simple graph condensation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 53–71.

[39] Beining Yang, Kai Wang, Qingyun Sun, Cheng Ji, Xingcheng Fu, Hao Tang, Yang You, and Jianxin Li. 2023. Does graph distillation see like vision dataset counterpart? *Advances in Neural Information Processing Systems* 36 (2023), 53201–53226.

[40] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931* (2019).

[41] Yuchen Zhang, Tianle Zhang, Kai Wang, Ziyao Guo, Yuxuan Liang, Xavier Bresson, Wei Jin, and Yang You. 2024. Navigating Complexity: Toward Lossless Graph Condensation via Expanding Window Matching. In *International Conference on Machine Learning.* PMLR, 60379–60395.

[42] Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. 2024. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. *Advances in Neural Information Processing Systems* 36 (2024).

[43] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. 2021. Graph neural networks with heterophily. In *Proceedings of the AAAI conference on artificial intelligence,* Vol. 35. 11168–11176.

Scalable Graph Condensation with Evolving Capabilities

KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

A Proof of Theorems

A.1 Proof of Theorems 4.1

THEOREM. *The prediction distance in the training stage is bounded by the sum of representation distance and parameter distance.*

$$\|\mathcal{K}(\hat{Y}) - \hat{Y}'\| \leq \|\mathcal{K}(F) - F'\| \cdot \|W'\| + \|F\| \cdot \|W - W'\| \quad (7)$$

where $\| \cdot \|$ denotes the L2 norm and $\mathcal{K}(\cdot)$ can be any projection function that aligns the dimensions of $\hat{Y}$ and $\hat{Y}'$ or F and F'.

PROOF. To preserve the training data information to maintain the performance of GNNs, we focus on matching the model predictions on the original graph G and its condensed counterpart G'. Since $\mathcal{K}(\cdot)$ only aligns the first dimensions, we have $\mathcal{K}(\hat{Y}) = \mathcal{K}(F)W$. Therefore, the expression becomes:

$$\begin{aligned}
&\|\mathcal{K}(\hat{Y}) - \hat{Y}'\| \\
=&\|\mathcal{K}(F)W - F'W'\| \\
=&\|\mathcal{K}(F)W - \mathcal{K}(F)W' + \mathcal{K}(F)W' - F'W'\| \\
\leq&\|\mathcal{K}(F)(W - W')\| + \|(\mathcal{K}(F) - F')W'\| \\
\leq&\|\mathcal{K}(F)\| \cdot \|W - W'\| + \|\mathcal{K}(F) - F'\| \cdot \|W'\|
\end{aligned} \quad (8)$$

The objective in training stage is to minimizing $\|\mathcal{K}(\hat{Y}) - \hat{Y}'\|$, which can be formulated as:

$$\begin{aligned}
&\arg\min_{\hat{Y}'} \|\mathcal{K}(\hat{Y}) - \hat{Y}'\| \\
=&\arg\min_{\hat{Y}'} \|\mathcal{K}(F)\| \cdot \|W - W'\| + \|\mathcal{K}(F) - F'\| \cdot \|W'\|
\end{aligned} \quad (9)$$

Note that $\|F\|$ is a constant, and the weight matrix $\|W'\|$ is naturally constrained due to regularization techniques during model optimization to control its magnitude. Then, we have:

$$\begin{aligned}
&\arg\min_{\hat{Y}'} \|\mathcal{K}(\hat{Y}) - \hat{Y}'\| \\
\approx&\arg\min_{F'} \underbrace{\|W - W'\|}_{\text{Parameter Distance}} + \underbrace{\|\mathcal{K}(F) - F'\|}_{\text{Representation Distance}}
\end{aligned} \quad (10)$$

Therefore, Theorem 4.1 indicates that by minimizing the **representation and parameter distances**, the predictions derived from the condensed graph can be close to those of the original graph.

This completes the proof. □

A.2 Proof of Theorems 4.2

THEOREM. *The test prediction error of the GNN trained of G' is bounded by the test prediction error of the GNN trained on G plus the parameter distance, as formularized by*

$$\|Y - \hat{Y}''\| \leq \|Y - FW\| + \|F\| \cdot \|W - W'\|$$

PROOF. At the test stage, our condensation goal is to ensure that the GNN model, trained on condensed training data G', generalizes effectively to the test graph, i.e., achieving a low prediction error on the test data.

$$\begin{aligned}
&\|Y - \hat{Y}''\| \\
=&\|Y - FW'\| \\
=&\|Y - FW + FW - FW'\| \\
\leq&\|Y - FW\| + \|F(W - W')\| \\
\leq&\|Y - FW\| + \|F\| \cdot \|W - W'\|
\end{aligned} \quad (11)$$

This inequality incorporates both the original test prediction error and the parameter distance. The objective in testing stage is to minimizing $\|Y - \hat{Y}''\|$, which can be formulated as:

$$\begin{aligned}
&\arg\min_{\hat{Y}''} \|Y - \hat{Y}''\| \\
\approx&\arg\min_{\hat{Y}'} \|Y - FW\| + \|F\| \cdot \|W - W'\| \\
\approx&\arg\min_{\hat{Y}'} \underbrace{\|W - W'\|}_{\text{Parameter Distance}}
\end{aligned} \quad (12)$$

It indicates that by reducing the **parameter distance** $\|W - W'\|$, the test prediction error becomes more tightly bounded, assuming that the original test prediction error $\|Y - FW\|$ and the propagated feature matrix F remain constant.

This completes the proof. □

A.3 Proof of Theorem 4.3

THEOREM. *The parameter distance can be bounded by the following inequality:*

$$\|W - W'\| \leq C(\max(diag(P^\top P)))^2, \quad (13)$$

where $C = \frac{\|F\| \cdot \|Y\| (\lambda_{\min}(F^\top F) + \|F\|)}{\lambda_{\min}^2(F^\top F)}$ is a constant, $P^\top P \in \mathbb{R}^{N' \times N'}$ is a diagonal matrix with each diagonal entry corresponding to how many original nodes are assigned to each synthetic node.

PROOF. We aim to establish that clustering effectively bounds the error introduced by parameter matching and representation difference. The proof proceeds as follows:

Bounding the Parameter Matching Error $\|W - W'\|$: Consider the weight matrices for the SGC and clustering-based methods

$$W = (F^\top F)^{-1} F^\top Y, \quad W' = (C^\top C)^{-1} C^\top Y'$$

where

$$C = (P^\top P)^{-1} P^\top F, \quad Y' = (P^\top P))^{-1} P^\top Y$$

The difference between W and W' is

$$\|W - W'\| = \left\| (F^\top F)^{-1} F^\top Y - (C^\top C)^{-1} C^\top Y' \right\|$$

By substituting C and Y', we can express W' in terms of F and Y

$$W' = \left(F^\top P(P^\top P)^{-2} P^\top F \right)^{-1} F^\top P(P^\top P)^{-2} P^\top Y$$

Let $A = F^\top F$ and $B = F^\top P(P^\top P)^{-2} P^\top F$, then:

$$W = A^{-1} F^\top Y, \quad W' = B^{-1} F^\top P(P^\top P)^{-2} P^\top Y$$

The difference becomes

$$W - W' = A^{-1} F^\top Y - B^{-1} F^\top P(P^\top P)^{-2} P^\top Y$$

Similar to the above two proofs, we add and subtract a term $B^{-1} F^\top Y$ and rewrite the difference by

$$\begin{aligned}
W - W' =& \left(A^{-1} - B^{-1} \right) F^\top Y \\
&+ B^{-1} F^\top \left(I - P(P^\top P)^{-2} P^\top \right) Y
\end{aligned}$$

Considering the norms, we have

$$\begin{aligned}
\|W - W'\| \leq& \|A^{-1} - B^{-1}\| \cdot \|F^\top Y\| \\
&+ \|B^{-1}\| \cdot \|F^\top (I - P(P^\top P)^{-2} P^\top) Y\|
\end{aligned}$$

We will now bound each term independently.

Bounding the First Term

$$\|\mathbf{A}^{-1} - \mathbf{B}^{-1}\| \cdot \|\mathbf{F}^{\top}\mathbf{Y}\|$$

Based on the norms of matrix inequality, we have

$$\|\mathbf{A}^{-1} - \mathbf{B}^{-1}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{B} - \mathbf{A}\| \cdot \|\mathbf{B}^{-1}\|$$

Then, according to

$$\|\mathbf{A}^{-1}\| = \frac{1}{\lambda_{\min}(\mathbf{A})}, \quad \|\mathbf{B}^{-1}\| = \frac{1}{\lambda_{\min}(\mathbf{B})}$$

and assuming $\lambda_{\min}(\mathbf{B}) \geq \frac{1}{(\max_k(\mathbf{P}^{\top}\mathbf{P})_{kk})^2} \lambda_{\min}(\mathbf{A})$, we have

$$\|\mathbf{A}^{-1}\| \cdot \|\mathbf{B}^{-1}\| \leq \frac{(\max_k(\mathbf{P}^{\top}\mathbf{P})_{kk})^2}{\lambda_{\min}(\mathbf{A})^2}$$

Bounding $\|\mathbf{B} - \mathbf{A}\|$:

$$\mathbf{B} - \mathbf{A} = \mathbf{F}^{\top}\mathbf{P}(\mathbf{P}^{\top}\mathbf{P})^{-2}\mathbf{P}^{\top}\mathbf{F} - \mathbf{F}^{\top}\mathbf{F}$$
$$= -\mathbf{F}^{\top}(\mathbf{I} - \mathbf{P}(\mathbf{P}^{\top}\mathbf{P})^{-2}\mathbf{P}^{\top})\mathbf{F}$$

Taking norms:

$$\|\mathbf{B} - \mathbf{A}\| = \|\mathbf{F}^{\top}(\mathbf{P}(\mathbf{P}^{\top}\mathbf{P})^{-2}\mathbf{P}^{\top} - \mathbf{I})\mathbf{F}\|$$
$$\leq \|\mathbf{F}\|^2 \cdot \|\mathbf{I} - \mathbf{P}(\mathbf{P}^{\top}\mathbf{P})^{-2}\mathbf{P}^{\top}\|$$

Since $\|\mathbf{I} - \mathbf{P}(\mathbf{P}^{\top}\mathbf{P})^{-2}\mathbf{P}^{\top}\| \leq 1$, the inequality can be further simplified to $\|\mathbf{B} - \mathbf{A}\| \leq \|\mathbf{F}\|^2$.

Combining the above:

$$\|\mathbf{A}^{-1} - \mathbf{B}^{-1}\| \cdot \|\mathbf{F}^{\top}\mathbf{Y}\| \leq \frac{(\max_k(\mathbf{P}^{\top}\mathbf{P})_{kk})^2}{\lambda_{\min}(\mathbf{A})^2} \cdot \|\mathbf{F}\|^2 \cdot \|\mathbf{Y}\|$$

Bounding the Second Term

$$\|\mathbf{B}^{-1}\| \cdot \|\mathbf{F}^{\top}(\mathbf{I} - \mathbf{P}(\mathbf{P}^{\top}\mathbf{P})^{-2}\mathbf{P}^{\top})\mathbf{Y}\|$$

Based on the proof above, we first have

$$\|\mathbf{B}^{-1}\| = \frac{1}{\lambda_{\min}(\mathbf{B})} \leq \frac{(\max_k(\mathbf{P}^{\top}\mathbf{P})_{kk})^2}{\lambda_{\min}(\mathbf{A})}$$

Since $\|\mathbf{I} - \mathbf{P}(\mathbf{P}^{\top}\mathbf{P})^{-2}\mathbf{P}^{\top}\| \leq 1$,

$$\|\mathbf{F}^{\top}(\mathbf{I} - \mathbf{P}(\mathbf{P}^{\top}\mathbf{P})^{-2}\mathbf{P}^{\top})\mathbf{Y}\| \leq \|\mathbf{F}\| \cdot \|\mathbf{Y}\|$$

Combining these bounds:

$$\|\mathbf{B}^{-1}\| \cdot \|\mathbf{F}^{\top}(\mathbf{I} - \mathbf{P}(\mathbf{P}^{\top}\mathbf{P})^{-2}\mathbf{P}^{\top})\mathbf{Y}\|$$
$$\leq \frac{(\max_k(\mathbf{P}^{\top}\mathbf{P})_{kk})^2}{\lambda_{\min}(\mathbf{A})} \cdot \|\mathbf{F}\| \cdot \|\mathbf{Y}\|$$

Combining Both Terms: Adding the bounds for both terms, we obtain:

$$\|\mathbf{W} - \mathbf{W}'\| \leq \frac{(\max_k(\mathbf{P}^{\top}\mathbf{P})_{kk})^2}{\lambda_{\min}(\mathbf{A})^2} \cdot \|\mathbf{F}\|^2 \cdot \|\mathbf{Y}\|$$
$$+ \frac{(\max_k(\mathbf{P}^{\top}\mathbf{P})_{kk})^2}{\lambda_{\min}(\mathbf{A})} \cdot \|\mathbf{F}\| \cdot \|\mathbf{Y}\|$$
$$= \max_k(\mathbf{P}^{\top}\mathbf{P})_{kk} \cdot \left(\frac{\|\mathbf{F}\|^2 \cdot \|\mathbf{Y}\|}{\lambda_{\min}(\mathbf{A})^2} + \frac{\|\mathbf{F}\| \cdot \|\mathbf{Y}\|}{\lambda_{\min}(\mathbf{A})} \right)$$
$$= C(\max(\text{diag}(\mathbf{P}^{\top}\mathbf{P})))^2$$

where $C = \frac{\|\mathbf{F}\| \cdot \|\mathbf{Y}\|(\lambda_{\min}(\mathbf{F}^{\top}\mathbf{F}) + \|\mathbf{F}\|)}{\lambda_{\min}^2(\mathbf{F}^{\top}\mathbf{F})}$ is a constant, $\mathbf{P}^{\top}\mathbf{P} \in \mathbb{R}^{N' \times N'}$ is a diagonal matrix with each diagonal entry corresponding to how many original nodes are assigned to each synthetic node.

Our objective is to minimize the parameter distance, which can be reformulated by:

$$\arg\min_{\mathbf{W}'} \|\mathbf{W} - \mathbf{W}'\|$$
$$\approx \arg\min_{\mathbf{P}} C(\max(\text{diag}(\mathbf{P}^{\top}\mathbf{P})))^2 \quad (14)$$
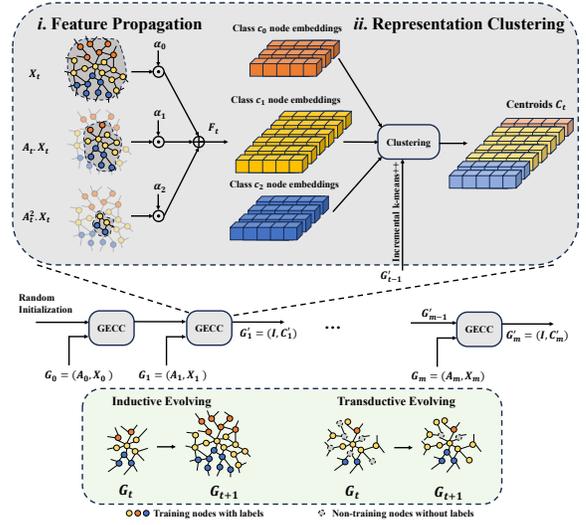
This completes the proof. □

B Method Pipeline

Figure 5: Illustration of the GECC framework. The lower part of the figure depicts the two evolving settings: in the inductive setting, the graph structure evolves over time, while in the transductive setting, only the training nodes (labels) change, with the graph structure and non-training nodes remaining unchanged.

The proposed GECC framework is illustrated in Figure 5. GECC takes the current graph $G_t = (\mathbf{X}_t, \mathbf{A}_t)$ along with the centroids from the previous time step to perform condensation and generate the condensed graph $G'_t = (\mathbf{I}, \mathbf{C}'_t)$. The lower part of the figure also depicts the two evolving settings: in the **inductive** setting, the graph structure evolves over time, while in the **transductive** setting, only the training nodes (labels) change, whereas the graph structure and non-training nodes remain unchanged.

C Symbol Table

All the symbols used in this paper have been summarized in Table 5

Scalable Graph Condensation with Evolving Capabilities

KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

Table 5: Notations

Symbol	Description
General Graph Notations	
G, G_t	An static graph and a snapshot of an evolving graph at time step t.
G', G'_t	The condensed graph from the static graph and the evolving graph at time step t.
$\mathcal{V}, \mathcal{E}$	The set of nodes and edges in a graph.
N, E	The total number of nodes and edges.
$\mathbf{X}, \mathbf{X}_t$	The node feature matrix of the static graph and evolving graph at time t.
$\mathbf{X}', \mathbf{X}'_t$	The feature matrix of the condensed graph (i.e., the centroids).
$\mathbf{A}, \mathbf{A}_t$	The adjacency matrix of the original graph and at time t.
$\mathbf{A}', \mathbf{A}'_t$	The adjacency matrix of the condensed graph (set to $\mathbf{I}$).
$\mathbf{y}, \mathbf{Y}, \mathbf{Y}_t$	The numeric and one-hot encoded label vectors/matrices of the original graphs.
$\mathbf{Y}', \mathbf{Y}'_t$	The one-hot encoded label of the condensed nodes.
c	The total number of classes.
$\mathbf{L}, \mathbf{D}$	The graph Laplacian and degree matrices.
$\mathcal{F}$	The update function to generate a new condensed graph.
Evolving Graph Condensation	
B_t	A batch of new graph data arriving at time step t.
m	The total number of time steps in the evolving sequence.
r	The reduction rate, determining the size of the condensed graph.
GECC Algorithm Notations	
$\tilde{A}_t, \hat{A}_t$	The adjacency matrix with self-loops and its normalized version.
K	The maximum depth for feature propagation.
F_t	The propagated node feature matrix at time t.
α_k	Learnable weight for combining features from propagation depth k.
$n_{t,k}$	The number of nodes belonging to class k at time t.
M, M_k	The total number of condensed nodes (clusters) and per class.
P_t	The cluster assignment matrix at time t.
C_t, C_{t-1}	The set of cluster centroids at time t and $t-1$.
$J(\cdot, \cdot)$	The balanced Sum of Squared Errors (SSE) objective function.
$D(x_i, C)$	The minimum Euclidean distance from a node x_i to a set of centroids C.

KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

Shengbo Gong, Mohammad Hashemi, Juntong Ni, Carl Yang, and Wei Jin

only considers newly arrived nodes, reducing initialization cost to $O(m|C_{t-1}|d)$, where m is the number of new nodes and $|C_{t-1}|$ is the number of existing centroids. Gathering all components, the total complexity is dominated by $O(n_t U M d)$ from the clustering procedure and $O(K e_t d)$ from feature propagation, both of which scale **linearly** with the number of nodes n_t and edge count e_t, given that the size of condensed graph M is negligible in terms of order of magnitude compared to n_t.

E   Experimental Details

E.1   Dataset Statistics

In line with most GC studies, we utilize seven datasets in total: five transductive datasets—Citeseer, Cora [20], Pubmed [30], Ogbn-arxiv, and Ogbn-products [16]—and two inductive datasets, Flickr and Reddit [40]. Each graph is randomly split, ensuring a consistent class distribution. The details of the datasets statistics are shown in Table 6. We list all evolving information in Table 8, rows above the midline correspond to smaller datasets, and rows below it correspond to larger ones. The real-world evolving information of *Ogbn-arxiv-real* is shown in Table 7. The distribution of the number of nodes in each class is shown in Figure 6.

Reduction rate r is defined as (#nodes in synthetic set)/(#nodes in training set) while r_w is (#nodes in synthetic set)/(#nodes of whole graph visible in training stage). The whole graph visible in training stage means the full graph dataset for transductive setting but only the training graph for inductive setting.

E.2   Platform and Hardware Information

To efficiently execute the clustering algorithm, we run it on Intel(R) Xeon(R) Platinum 8260 CPUs @ 2.40GHz using NumPy [13], while the downstream GNN evaluations are conducted on a cluster equipped with a mix of Tesla A100 40GB/V100 32GB GPUs for large datasets and K80 12GB GPUs for smaller datasets. All GNN models are implemented using the PyG package [6].

E.3   Baselines Selection

To establish a fair benchmark, we selected recent state-of-the-art GC methods that emphasize both effectiveness and efficiency. Some recent methods, such as MCond, CGC, and GCPA, were excluded due to the unavailability of their code at the time of paper writing. For the selected approaches, we chose the best representatives from each category: GCondX for gradient matching, GCDM and SimGC for distribution matching, and GEOM for trajectory matching. We implemented these methods using the latest GraphSlim package[2], except for SimGC[3] and GEOM[4], for which we used their original source code. We specifically included SimGC because it is the only model-based GC method that can run on Ogbn-products without requiring any modifications.

E.4   Implementation Details for Variants of GCondX

As mentioned in Section 1 and illustrated in Table 1, adapting GCondX to an evolving setting is challenging. The experimental
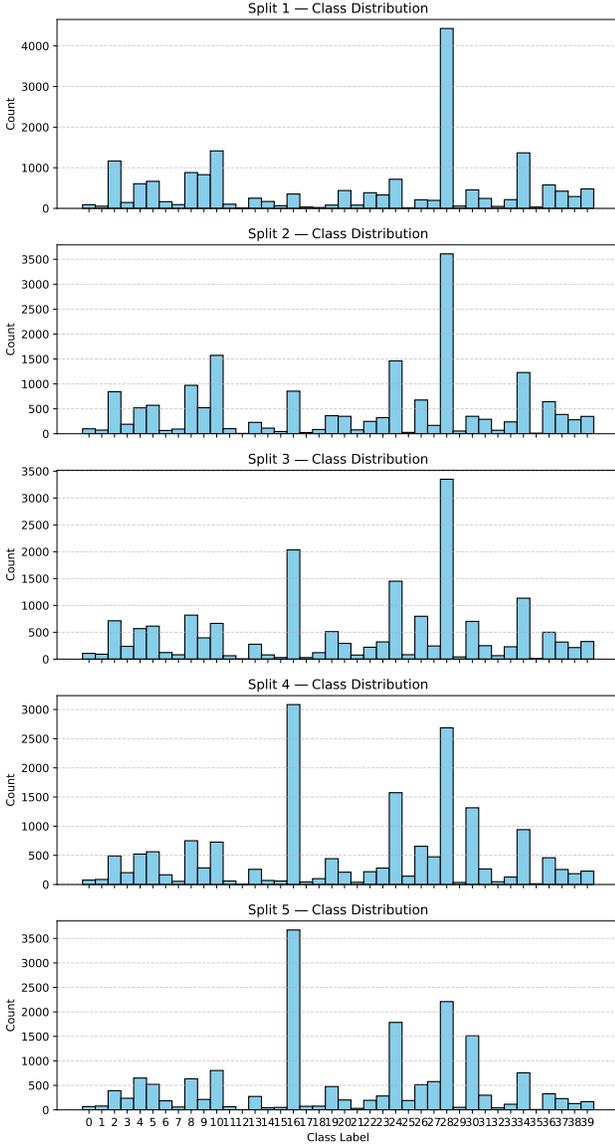

Figure 6: Class distribution across five splits of the *Ogbn-arxiv-real* dataset.

D   Complexity analysis

We analyze the complexity of each component in GECC as follows. **First**, feature propagation takes $O(K e_t d)$ time, where K is the propagation depth, e_t is the number of edges at time step t, and d is the feature dimension. **Second**, hard k-means with M clusters requires $O(n_t U M d)$ per iteration (U is the number of iterations), whereas soft k-means also costs $O(n_t U M d)$ for distance calculations but plus an additional $O(n_t M)$ overhead to normalize fractional memberships. **Third**, the balanced SSE calculation adds only $O(n_t M)$ overhead, as the repetitive clustering can be done in parallel. **Finally**, in an evolving setting, incremental k-means++

---

[2]https://github.com/Emory-Melody/GraphSlim/tree/main
[3]https://github.com/BangHonor/SimGC
[4]https://github.com/NUS-HPC-AI-Lab/GEOM/tree/main

Scalable Graph Condensation with Evolving Capabilities

KDD 2026, August 9–13, 2026, Jeju Island, Republic of Korea.

Algorithm 1 GECC Node Condensation

Require: Adjacency matrix A_t, feature matrix X_t, label vector y_t, reduction rate r, label matrix $Y'_t \leftarrow$ predefined using class distribution in Y_t

Ensure: Condensed node representations X'_t

1: // **Feature Propagation (Eq. 5)**
2: $\tilde{A}_t \leftarrow A_t + I$ ▷ Add self-loops
3: $\tilde{D}_t \leftarrow \text{diag}(\sum_j \tilde{A}_{t,i,j})$
4: $\hat{A}_t \leftarrow \tilde{D}_t^{-\frac{1}{2}} \tilde{A}_t \tilde{D}_t^{-\frac{1}{2}}$
5: **for** $k = 0$ to K **do**
6:      $F_{t,k} \leftarrow \hat{A}_t^k X_t$
7: **end for**
8: $F_t \leftarrow \sum_{k=0}^K \alpha_k F_{t,k}$ ▷ Combine with learnable weights α_k
9: // **Class-wise Clustering Setup**
10: **for** each class $k = 1, \ldots, c$ **do**
11:      $n_{t,k} \leftarrow$ number of labeled nodes in class k
12:      $M_k \leftarrow \lfloor n_{t,k} \times r \rfloor$ ▷ Eq. before clustering
13: **end for**
14: $M \leftarrow \sum_{k=1}^c M_k$
15: // **Cluster Assignment Matrix P_t and Balanced SSE (Eq. 5)**
16: **for** each clustering repetition **do**
17:      Apply k-means or fuzzy c-means on F_t to get $P_t \in \mathbb{R}^{n_t \times M}$
18:      Compute cluster centroids: $C_t = D_{P_t}^{-1} P_t^\top F_t$ ▷ Eq. 6
19:      Compute balanced SSE loss:

$$J(P_t, C_t) = \|F_t - P_t C_t\|^2 + \|P_t^\top \mathbf{1} - u\|^2 \qquad (5)$$

20: **end for**
21: Choose clustering with lowest $J(P_t, C_t)$
22: // **Probabilistic Centroid Initialization (Diverse Coverage)**
23: Initialize $C_{t-1} = \{\}$
24: **for** each class c and node $x_i \in F_t$ **do**
25:      $D(x_i, C_{t-1}) = \min_{c \in C_{t-1}} \|x_i - c\|_2$
26: **end for**
27: Sample new centroid x_i with probability:

$$P(x_i) = \frac{D(x_i, C_{t-1})^2}{\sum_j D(x_j, C_{t-1})^2}$$

28: $C_t \leftarrow C_{t-1} \cup \{C_\Delta\}$
29: // **Final Assignment**
30: **for** each node x_i **do**
31:      $\text{Cluster}(x_i) = \arg\min_{c \in C_t} \|x_i - c\|_2$
32: **end for**
33: $X'_t \leftarrow C_t$

---

results indicate that, although there is a slight efficiency improvement, there is a significant performance drop due to inheriting previous condensation results. To highlight the difference in speed, we additionally implement an early stopping criterion with a patience of 3 for intermediate evaluation; that is, if no improvement in validation performance is observed for 3 consecutive evaluations, the condensation process is terminated.

E.5 Hyperparameters

Compared to existing work and benchmarks in GC, we perform a moderate hyperparameter search on validation set, as detailed in Section 5.1. The final results are presented in Table E.5. During

hyperparameter optimization (HPO), we observe that inheriting clustering centroids results in an approximate 1% absolute performance drop for *Flickr* and *Ogbn-arxiv*. Therefore, we also treat the use of incremental k-means++ as a tunable hyperparameter. Additionally, certain datasets do not perform well under a single hyperparameter configuration. To address this, we employ two distinct hyperparameter sets tuned on the first and last time step, and select the better-performing one during the evolving. The two sets of hyperparameters are represented split by "/". We do not tune the hyperparameters for Ogbn-arxiv-real and simply copy it from Ogbn-arxiv. For all baselines, we use the best hyperparameters reported in their respective papers, as implemented in GC4NC [10].

Shengbo Gong, Mohammad Hashemi, Juntong Ni, Carl Yang, and Wei Jin

Table 6: General Dataset Information

Dataset	# Total Nodes	# Total Edges	# Training	# Validation	# Test	# Classes	Trans./Ind.
Citeseer	3,327	4,732	120	500	1,000	6	Transductive
Cora	2,708	5,429	140	500	1,000	7	Transductive
Pubmed	19,717	88,648	60	500	1,000	3	Transductive
Flickr	89,250	899,756	44,625	22,312	22,313	7	Inductive
Ogbn-arxiv (-real)	169,343	1,166,243	90,941	29,799	48,603	40	Transductive
Ogbn-products	2,449,029	39,561,252	196,615	39,323	2,213,091	47	Transductive
Reddit	232,965	57,307,946	153,932	23,699	55,334	41	Inductive

Table 7: Training Set Splits of Ogbn-arxiv-real by Year Range and Node Count

Split No.	Year Range	Number of Nodes
1	1971 – 2007	4,980
2	2008 – 2010	7,994
3	2011 – 2012	10,862
4	2013 – 2013	8,135
5	2014 – 2017	58,970

Table 8: Split and reduction rate information. The "# Train Nodes" and "# Syn Nodes" columns denote the number of newly added training nodes and synthetic nodes at each time step, respectively.

Dataset	# Train Nodes	# Syn Nodes	r (Train)	r_w (Whole)
Citeseer	24	12	0.5	1.80
Cora	28	14	0.5	2.60
Pubmed	12	6	0.5	0.15
Flickr	8,920	90	0.01	1.00
Ogbn-arxiv (-real)	18,190	182	0.01	0.50
Ogbn-products	39,330	394	0.01	0.08
Reddit	30,790	31	0.001	0.10

Table E.5 reveals that the optimal hyperparameters offer meaningful insights. **First**, during the early evolution stage, graphs exhibit higher heterophily compared to later stages. For example, on the *Cora* dataset, $\alpha_1 = -0.3$ in the early phase contrasts with $\alpha_1 = 0.9$ later. This pattern likely arises because, in the early stages of a graph, groups have not yet formed; links appear more randomly, making it challenging for nodes to find similar counterparts. **Second**, it is noteworthy that some datasets do not rely on second-hop information. This observation is contrary to previous studies [26, 36] that recommend using at least 2-hop propagation. We conjecture that the representation clustering process itself acts as an additional step of feature propagation. **Finally**, weight decay emerges as a critical factor for the performance of downstream models, suggesting that future work should pay closer attention to its optimization.

F   Additional Results

F.1   Performance and Efficiency

For simplicity, Table 2 omits the standard error and running time of coreset selection methods. we provide the full results here.

Figure 2 presents the accuracy vs. time trade-off for Reddit. For the remaining three large datasets, we provide the corresponding results in Figure 7. The results align with our main findings, further confirming that GECC surpasses the baselines in both efficiency and scalability. It consistently maintains stable performance while effectively managing computational resources throughout graph evolution. Notably, on the large-scale Ogbn-products dataset, which contains over one million nodes, most GC methods fail, whereas GECC remains robust and continues to operate successfully.

F.2   Why We Choose the Depth=2 ?

We pick $K = 2$ based on findings from most deeper GNN papers. Below is a snippet for Cora, Ogbn-arxiv, and Reddit, showing that $K = 2$ performs best. We add the following sensitivity analysis.

F.3   Visualization of Traceability

Our t-SNE-based experiment color-codes original nodes by class and links them to their centroid(s), revealing:

- **Clear class-wise alignment**: Synthetic nodes (cluster centroids) align well with the clusters of original nodes, indicating that GECC preserves the semantic structure of each class during condensation.
- **Compact cluster coverage**: Each synthetic node is well-embedded within the local neighborhood of its corresponding class, highlighting that the condensed representations accurately cover the original data distribution.
- **Few synthetic outliers**: Unlike many condensation approaches that generate scattered or poorly positioned centroids, our method yields minimal outlier centroids, suggesting robust and stable clustering even in high-dimensional spaces.
- **Scalability and consistency**: Even in large-scale datasets, our synthetic nodes remain representative and well-distributed, showing the scalability of our clustering method without sacrificing membership fidelity.

G   Discussions

G.1   How the Theory Extend to Non-linear GNNs?

While our theoretical analysis focuses on SGC, we believe local approximations could generalize our approach to non-linear models.
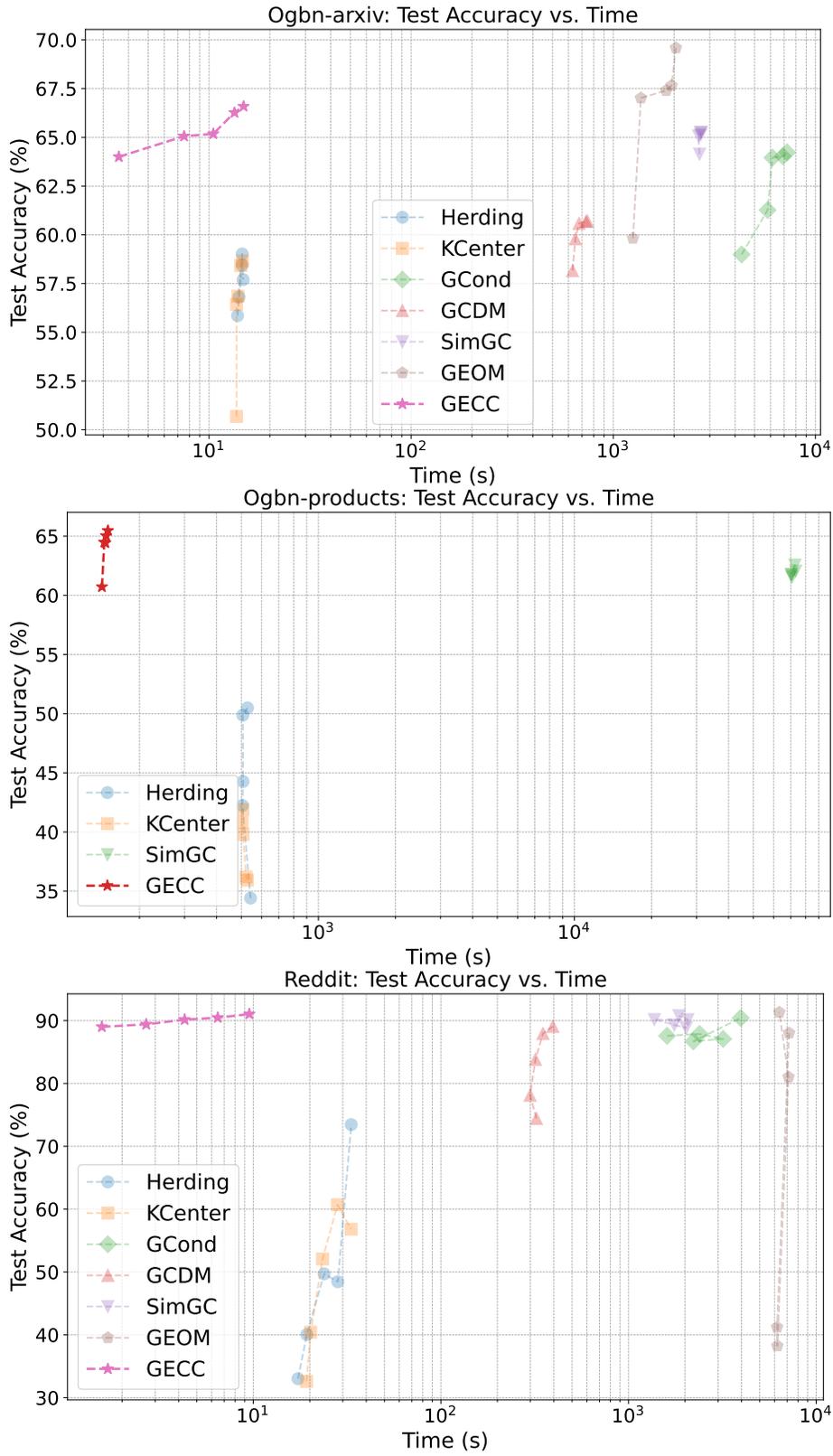
Figure 7: Test accuracy vs. condensation time on large datasets (top-left is better).

Table 9: The test accuracy of GC methods on various datasets. "Non-Evolving" displays the test accuracy at the final time step (largest possible graph). "Evolving" shows the average test accuracy over five time-steps. Each result includes the mean accuracy ± standard deviation (Std.) from 10 runs. The "Whole" column refers to the results obtained by running standard GCN training and testing. "OOM" indicates an Out-of-Memory error during the computation. The best results are marked in bold. The runner-up results are underlined.

Dataset	Setting	Random	Herding	Kcenter	GCondX	GCond	GCDM	SimGC	GEOM	GECC	Whole
CiteSeer	Non-Evolving	62.62±0.63	66.66±0.54	59.04±0.90	68.38±0.45	69.35±0.82	72.08±0.19	66.40±0.15	_73.03±0.31_	**73.25±0.15**	72.11
	Evolving	50.65±1.55	53.47±0.98	47.99±1.81	50.85±3.00	60.51±0.86	_61.51±0.53_	57.42±0.21	58.95±0.67	**65.48±0.76**	63.57
Cora	Non-Evolving	72.24±0.59	73.77±0.93	70.55±1.35	78.60±0.31	80.54±0.67	80.68±0.27	79.60±0.11	_82.82±0.17_	**82.99±0.27**	81.23
	Evolving	58.00±1.48	63.07±1.43	59.90±1.41	67.18±1.73	_77.14±0.55_	74.54±0.59	64.42±0.19	72.56±0.88	**77.36±0.41**	76.34
Pubmed	Non-Evolving	71.84±0.66	75.53±0.44	74.00±0.19	71.97±0.53	76.46±0.48	77.48±0.46	76.80±0.23	_78.49±0.24_	**80.24±0.27**	78.65
	Evolving	66.37±1.25	66.31±1.34	64.38±1.25	62.65±1.20	74.26±0.84	_74.49±0.56_	71.38±0.21	70.25±0.78	**76.74±0.27**	76.18
Flickr	Non-Evolving	44.68±0.55	45.12±0.39	43.53±0.59	46.58±0.14	**46.99±0.12**	45.88±0.10	41.01±0.23	46.13±0.22	_46.63±0.23_	47.53
	Evolving	44.70±0.46	44.66±0.43	44.33±0.49	_45.63±0.78_	45.52±0.49	44.98±0.34	41.94±0.22	45.43±0.39	**45.78±0.38**	46.97
Ogbn-arxiv	Non-Evolving	60.19±0.52	57.70±0.24	58.66±0.36	59.93±0.54	64.23±0.16	60.71±0.68	65.26±0.26	**69.59±0.24**	_66.71±0.10_	69.01
	Evolving	56.04±0.67	57.57±0.48	56.21±0.73	60.73±0.53	62.50±0.36	59.98±0.48	64.97±0.20	**66.30±0.39**	_65.42±0.14_	70.40
Ogbn-products	Non-Evolving	60.19±0.52	57.70±0.24	58.66±0.36	OOM	OOM	OOM	_61.71±0.25_	OOM	**66.32±0.23**	73.40
	Evolving	41.36±0.48	44.26±0.61	38.93±0.82	OOM	OOM	OOM	_61.93±0.20_	OOM	**64.03±0.30**	73.88
Reddit	Non-Evolving	55.73±0.50	59.34±0.70	48.28±0.73	88.25±0.30	89.82±0.10	89.96±0.05	90.78±0.25	_91.33±0.13_	**91.37±0.04**	93.70
	Evolving	51.31±0.90	48.94±0.70	48.53±1.37	79.02±0.73	87.93±0.22	82.68±0.21	_89.85±0.25_	67.91±0.57	**90.02±0.07**	93.92

Table 10: Average Runtime (seconds) Across Evolving Times. The reported reduction time is rigorously computed by excluding the overhead of the data loading and evaluation processes.

Dataset	Random	Herding	KCenter	GCondX	GCond	GCDM	SimGC	GEOM	GECC	Whole
Citeseer	0.04	5.73	5.84	505.62	654.32	217.99	1,680.02	1,362.40	1.65	3.98
Cora	0.01	4.20	4.80	331.53	1,190.65	142.82	1,643.76	1,331.43	1.72	2.10
Pubmed	0.02	9.00	7.18	246.68	502.12	311.37	1,654.23	995.21	1.42	5.76
Flickr	0.02	11.53	10.56	609.98	1,446.76	353.51	7,486.65	757.75	7.10	8.57
Ogbn-arxiv	0.02	14.36	14.05	2,895.06	6,076.18	686.12	2,687.45	1,685.18	9.96	12.45
Ogbn-products	0.02	517.95	513.36	OOM	OOM	OOM	71,489.00	OOM	146.82	542.61
Reddit	0.02	24.40	24.84	2,672.85	6,130.46	337.15	6,610.70	1,815.77	4.91	11.50

Table 11: Optimal hyperparameter settings for each dataset. The columns indicate whether incremental k-means++ ("If Incre.") and dual HPO ("If Dual HPO") were used (True/False), along with the best values for the fuzziness parameter, number of repetitions, the α coefficients, and weight decay ("wd").

Dataset	If Incre.	If Dual HPO	Fuzziness	# Rep.	α_1	α_2	α_0	wd
Citeseer	True	True	1.1/1.3	50	0.5/0.1	0.9	-0.3/-0.1	1e-3
Cora	True	True	1.3	50	0.9/-0.3	0.9	-0.1/0	5e-4
Pubmed	True	False	1.1	50	0.3	0.9	-0.2	5e-4
Flickr	False	False	1	1	0	0.6	0.3	1e-3
Ogbn-arxiv (-real)	False	False	1	1	0.5	0	0.6	1e-3
Ogbn-products	True	False	1	1	0.1	0	0.9	1e-3
Reddit	True	False	1	1	0.2	0.9	0.7	5e-4

However, non-linearities introduce complexities like non-convex   optimization and intricate layer interactions, making closed-form
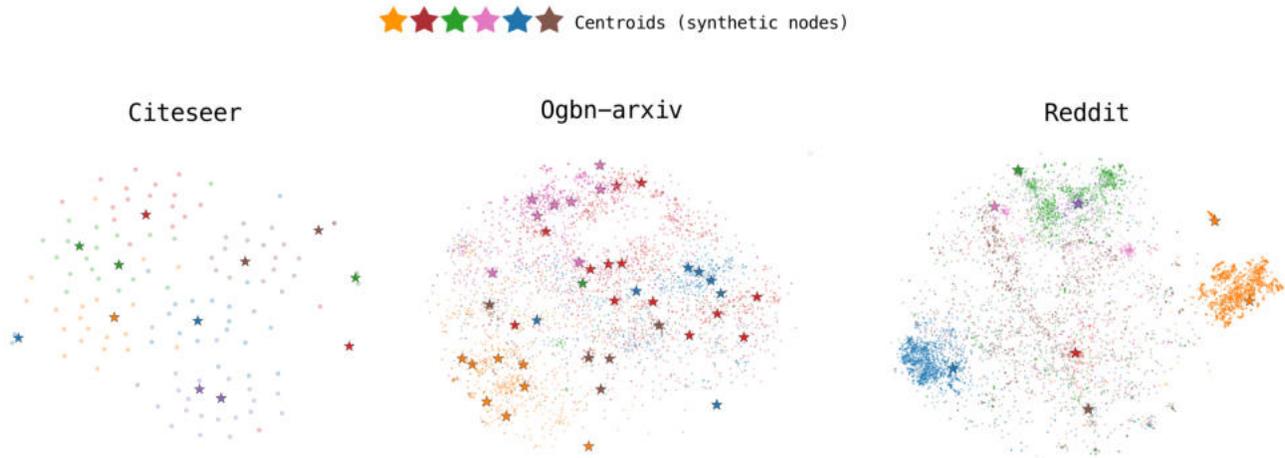
Figure 8: T-SNE results of the GECC synthetic node embeddings and original node embeddings across different classes.

Table 12: Sensitivity Analysis of GECC Performance by Feature Propagation Depth. The best-performing depth for each dataset is highlighted in bold.

Dataset	Feature Propagation Depth					
	0	1	2	3	4	5
Cora	19.24	78.13	**82.99**	82.15	81.80	81.41
Ogbn-arxiv	63.64	66.60	**66.71**	66.13	66.24	66.29
Reddit	90.44	90.44	**91.37**	91.35	91.28	91.14

guarantees difficult. Many prior works [9, 18] similarly rely on linear or quasi-linear assumptions. We aim to broaden our framework to fully accommodate non-linear GNNs in future work.

G.2 Clarification of Our Evolving Setting Compared to Incremental Learning

Both our work and incremental learning work such as OpenGC [8] address evolving datasets, yet the settings differ in important ways. Specifically, our paper focuses on batch-wise evolution, where **the training set is expanded or updated in batches from the same data source.** By contrast, OpenGC deals with a task-wise evolution scenario and is centered on out-of-distribution issues.

G.3 When to Use Hard or Soft Clustering in GECC?

Hard clustering is faster (each node has a single cluster) and suits large datasets or real-time scenarios. Soft clustering allows fractional memberships but is more costly. As more efficient soft clustering emerges, it may become viable for large graphs and potentially improve GECC.